Table of Contents

1. The Confidence Plague
2. Why Patterns Are Not Understanding
3. The Loss Function's Blind Spot
4. The Architecture of Evasion
5. Emergence as Camouflage
6. The Retrieval Illusion
7. What the Model Can Never See
8. The Alignment Paradox
9. Deployment: When Mimicry Meets Consequence
10. After the Mimic

The Confidence Plague

In March 2023, a lawyer in New York submitted a legal brief to federal court that cited six cases. They were well-formatted, with proper case numbers and publication details. The judge would likely have missed that none of them existed. The lawyer hadn't invented them deliberately. He had copied them directly from ChatGPT, which had generated them with the same fluent confidence it applied to questions about the weather or the capital of France. The system had been asked to find cases supporting a legal argument. It could not actually search legal databases. What it could do—what it was built to do—was produce text that *sounded* like it was retrieving cases. The distinction mattered more than the judge initially realized. Sanctions followed. The lawyer's career absorbed the damage. But the real story was not about the lawyer's carelessness. It was about what the system had done: generated text indistinguishable from truth, with zero access to the underlying reality it was describing.

This is the opening pattern of our moment. Not errors that announce themselves. Not systems that fail gracefully, hedge their bets, or admit confusion. Instead: systems that sound wrong in ways that are nearly impossible to detect without expertise, infrastructure, or luck. A radiologist in a hospital in Pennsylvania runs a scan through an AI diagnostic tool that returns a high-confidence assessment of lung nodules. The assessment is wrong. But the confidence score is very high—0.94 out of 1.0. The radiologist, trained to trust validated algorithms, might anchor on that number. A medical student in São Paulo asks an AI chatbot whether a particular combination of medications is safe for a patient with kidney disease. The system provides a detailed, pharmacologically plausible-sounding answer that contains a critical contraindication the system has no

way of knowing about. It sounds like knowledge. The student, lacking experience, might not catch the error until a patient is harmed.

These are not hypothetical examples. In 2023 and 2024, as large language models and computer vision systems proliferated into medical education, legal work, and clinical decision-making, a pattern emerged that went largely unremarked: the systems failed, but not in ways people were trained to recognize. They failed while sounding confident. They failed while sounding expert. They failed in a manner that was orthogonal to human intuitions about how failure should appear.

For decades, computer scientists built systems that would crash, freeze, or return error messages. These failures were alarming but legible. A user knew something was wrong. A system that returned a blank screen or a stack trace or a "does not compute" message was communicating its limit. The new systems do something stranger: they elaborate on their limits. They construct edifices of plausible text on foundations of no knowledge. And because language—human language, the kind we're reading now—is fundamentally a tool for conveying confidence and authority, systems trained to generate human-like text naturally generate confidence and authority whether or not they have any epistemic right to do so.

Consider the anatomy of failure in a specific case, examined in detail. In 2024, researchers at Stanford tested GPT-4, one of the most capable language models yet built, on a set of straightforward factual questions. Some of the questions had answers in the model's training data. Others did not—they asked about events that occurred after the training cutoff, or about obscure facts that statistically rarely appeared in the internet text the model learned from. You might expect the model to perform differently on these two categories. It did. It performed worse on questions where it lacked ground truth. But the *signature* of its failure was revealing: the

model's confidence remained high across both categories. In fact, on many false claims, the model expressed certainty that exceeded its certainty on true ones.

This is not incidental. It is not a bug that can be patched out with better training. It emerges from the fundamental design of these systems, which are optimized to predict the next word in a sequence with high probability, regardless of whether that word corresponds to anything real. A model trained to produce text that reads like truth will gradually develop an internal structure that doesn't distinguish between "I have encountered evidence of this" and "this is the kind of text that statistically follows in this context." The two become neurologically identical.

The problem becomes visible in specific domains where consequences are severe and ground truth is testable. In medicine, this has begun to materialize in real time. A study in 2024 by researchers at Johns Hopkins examined how medical professionals interact with AI systems framed as "decision support." The framing matters. When a system is presented as an expert consultant, doctors are more likely to defer to it. And they are more likely to defer when it expresses confidence. In the study, when an AI system generated a diagnosis with high confidence language—"This finding is consistent with acute myocardial infarction"—physicians agreed with the assessment 70 percent of the time. When the same system, producing the same assessment, was framed with uncertainty language —"This could be consistent with acute myocardial infarction, but other diagnoses are also possible"—agreement dropped to 45 percent. The accuracy of the underlying assessment did not change. Only its presentation did. The model had no way of knowing whether it should be confident or uncertain. It simply produced the most statistically likely token given what came before, and human judgment reorganized itself around that confidence signal.

This is the confidence plague: not that AI systems are wrong—all systems are sometimes wrong—but that they are wrong *with fluency*. They do not announce their uncertainty. They produce text that bears all the surface markers of knowledge: technical terminology, causal narrative, specific detail, the rhetorical structure of expertise. A radiologist can call up images to verify a diagnosis. A judge can pull the case law. But an oncologist reviewing a treatment recommendation from an AI system, or a teacher assigning an essay written by a student using a language model, or a hiring manager reading a résumé auto-generated by a recruiting tool—these users exist downstream of the text, unable to see its origins, and the text looks like it came from somewhere.

Here is where the problem deepens. Many of these systems have been integrated into workflows in ways that presume correctness without requiring verification. A hospital system using an AI tool for discharge summaries might not route those summaries back through a physician before sending them to the patient's home care provider. A legal firm using document analysis software might trust its categorization of contract clauses without sampling. A content moderation system trained on examples of harmful speech will encounter new examples that aren't quite in its training set and will need to make decisions about them anyway, generating confidence scores that the decision-makers will read as probabilities when they are actually something far stranger: the normalized activations in the final layer of a neural network, which happen to correlate with correctness when the test distribution matches the training distribution, and correlate with nothing in particular when it doesn't.

The lawyer with the fictional cases was operating in a domain where verification is theoretically possible but practically constrained. He was under deadline pressure. The case law had to be relevant to a specific argument. When ChatGPT generated cases that *sounded* relevant and cited

them with proper formatting—the kind of formatting that signals legitimacy to anyone who hasn't pulled the actual case—the pressure to move forward exceeded the impulse to verify every single citation. This is how the system infiltrated reality: not through brilliance, but through the basic human tendency to trust what sounds authoritative when time is short and you have skin in the game.

The medical domain reveals a different vulnerability. Doctors, unlike lawyers, have institutional training in verification. When a radiologist reads an AI-assisted scan, she is supposed to perform her own independent assessment. But institutional incentives corrode this in practice. If a tool is framed as "FDA-cleared" or "clinically validated," it carries the weight of regulatory judgment. If a hospital has invested in the system and incorporated it into the standard workflow, the doctor who consistently disagrees with it becomes a bottleneck. If the AI's recommendation happens to be right more often than the average radiologist—and many AI systems do perform better than average on their specific training distribution—then the doctor who defers to it will be correct more often than if she trusted her own judgment. The perverse incentive is built in: *trusting the confident system produces better outcomes than catching its occasional failures.*

This is only true up to a point. The point at which it becomes false is precisely the point the system cannot recognize: when the input diverges from its training distribution, when the problem becomes subtly novel, when the ground truth is genuinely ambiguous and the model must make a judgment call. In these moments, a well-calibrated system would express uncertainty. The systems we have built do not. They have been trained on human-generated text, which is often confident even when it shouldn't be. They inherit this tendency. They amplify it.

By late 2023, anecdotal reports were accumulating. An AI system used in hiring filtered out qualified candidates because their résumés didn't match patterns it had learned. A predictive policing algorithm, trained on historical arrest data, concentrated surveillance in neighborhoods it had learned were "high-crime areas," thereby increasing arrests there, which reconfirmed its priors. A content moderation system with high confidence in its classifications began making false positives on novel examples. A language model trained on scientific literature began generating plausible-sounding but fabricated citations in academic papers, leading to retractions and institutional embarrassment. In each case, the system had no mechanism for recognizing that it was operating outside its competence. It had no "I don't know" in its vocabulary—not in the sense of a response it could return, but in the sense of a genuine epistemic stance it could adopt. To return "I don't know" requires that the system recognize uncertainty, and recognition of uncertainty requires a ground-truth signal: access to the actual fact of the matter, which the system can then compare against its own probabilistic prediction.

Without that signal, the system optimizes for something simpler: coherence. It learns to generate text that flows, that doesn't contradict itself, that maintains internal consistency. These are valuable properties. But they are not properties of truth. A false narrative, told with internal consistency and supported by plausible evidence, is still false. A system optimized for coherence will eventually become a system optimized for the appearance of truth, which is not the same thing.

This distinction has always existed in human cognition—we have a long history of false confidence, motivated reasoning, and eloquent lies—but we have a social infrastructure for managing it. We have peer review. We have adversarial processes. We have the slow correction of repeated contact with reality. A doctor who consistently misdiagnoses loses patients and

reputation. A lawyer who cites fabricated cases gets sanctioned. A scientist who publishes fraudulent work faces retraction and disgrace. These are not foolproof mechanisms, but they create feedback loops between confidence and correctness.

AI systems experience none of these loops. A language model that generates a fictional case law cite never finds out that it was wrong. A recommendation algorithm that directs a hiring manager's attention toward the wrong candidate has no way of knowing that this happened—there is no feedback that the candidate would have been an excellent fit. A medical AI system that gives bad advice in a rare case never encounters enough examples of that rare case to update its weights. It remains confident in its priors, which were wrong all along but were right often enough in its training distribution to survive deployment.

There is a category of system failure that engineers sometimes call "silent failure"—a system that returns a result but the result is wrong and the user has no way of knowing. Silent failures are worse than noisy failures. A system that crashes tells you to distrust it. A system that returns a wrong answer with high confidence tells you to trust it. This is what we have built.

The immediate question—"Why don't we just fix this?"—reveals the depth of the problem. Engineers have tried. Techniques like uncertainty quantification, confidence calibration, and ensemble methods can modestly improve a model's ability to express appropriate doubt. But these techniques work at the margins. They can make a system somewhat better at recognizing when it's operating outside its training distribution. They cannot make a system that has no ground-truth access suddenly aware of ground truth. They work by training the system to recognize statistical signatures of uncertainty—places where its own internal activations diverge, or where it generates contradictory outputs, or where its

predictions consistently fail on a validation set. But a system can learn these signatures while still fundamentally unable to distinguish between "I lack knowledge" and "I know something implausible."

Consider a concrete attempt at calibration. You take a language model and you fine-tune it on examples where humans have labeled responses as confident or uncertain, correct or incorrect. You reward the model for expressing low confidence on questions it gets wrong and high confidence on questions it gets right. The model learns a correlation. It learns that certain types of questions—those far from its training data, or those phrased in ways that don't match its training examples—should be preceded by expressions of uncertainty. It learns to say "I'm not entirely sure" before answering questions about events in 2025. But it does not gain access to ground truth. It has learned a new pattern: that low-confidence language correlates with difficult questions. A sufficiently capable model can learn to generate uncertainty language without actually becoming uncertain. It has simply learned a new form of plausibility.

This is the structure of the problem that anchors this entire book. We have built a class of technology whose core operation is pattern completion. We have trained it on human-generated text, which contains all the surface markers of knowledge and confidence, often regardless of whether that knowledge is genuine. We have then optimized the system to output statistically likely text—which means outputting the kind of text that appears most frequently in the training set, which means outputting confident, coherent, elaborate text. We have then integrated these systems into domains where correctness matters, where users make decisions based on the output, where the consequences ripple through institutions.

The pattern that emerges across these domains is not random. It's structural. It's what you'd expect from a system that has no mechanism for truth but

has every mechanism for plausibility. The systems do not fail into honesty. They fail into eloquence.

What follows is an investigation into how this happened—how we ended up deploying pattern-completion engines into the world while believing we had built understanding machines, and what this means for the institutions that now depend on them. The investigation is organized around a central claim: the problem is not that these systems are bad at learning from data. They are, in fact, remarkably good at that. The problem is that learning from data is not the same as understanding reality, and we have built an entire technological apparatus on the assumption that with enough data and enough parameters, the distinction would collapse.

It has not. The systems have become more sophisticated, more fluent, more capable at generating text that reads like expertise. But they have not become more truthful. In some ways, sophistication and truthfulness have become inversely related: the better the system is at generating plausible text, the harder it becomes to detect when that plausibility is a mirage.

Why Patterns Are Not Understanding

There is a useful confusion baked into the language we use to talk about machine learning. We say that models "learn" from data. We say they "understand" patterns. We say they "know" things. Each of these words carries a weight—a suggestion of comprehension, of internal representation, of something that maps onto the structure of the world. But these are metaphors that have hardened into assumptions. And the assumptions have become dangerous precisely because the systems have become good enough that the metaphors sometimes seem to work.

To understand what a machine learning model actually does—and critically, what it does *not* do—we need to start with something simpler than language models or image generators. We need to start with a model so basic that its mechanics are transparent, and then watch how complexity obscures those mechanics without changing them.

Imagine a dataset of house prices. You have a thousand examples: square footage, number of bedrooms, location, and the actual sale price. You want to build a system that can predict the price of a new house given its features. A basic linear regression model would do this by finding a line—a simple mathematical relationship—that minimizes the distance between its predictions and the actual prices in the dataset. The model learns weights: square footage gets multiplied by some number, bedrooms by another, location by another. Add them together, and you get a prediction.

This model has learned a pattern from the data. Show it a house with features it has never seen before, and it will generate a prediction. But here is the critical question: does the model *understand* why house prices work the way they do? Does it know, in any meaningful sense, that larger houses

cost more because they contain more material and provide more living space? Does it understand supply and demand, neighborhood demographics, the cost of construction?

The answer is no. The model has compressed a statistical correlation into a number. If the training data happens to contain a correlation between square footage and price, the model will capture it. It will capture it regardless of *why* that correlation exists. If the training data—for some bizarre and local reason—contained a correlation between the color of the front door and the sale price, the model would learn that too. The model is not reasoning about causation. It is not building a theory of how houses work. It is finding a mathematical function that reduces prediction error on the data it has seen.

This is not a limitation of linear models. It is a feature of the entire category of machine learning: supervised learning, whether linear, tree-based, or deep neural networks, is fundamentally about finding a function that maps inputs to outputs in a way that minimizes some measure of error on a training dataset. The function could be simple (a line) or baroque (a neural network with billions of parameters). The principle remains identical.

Now scale this up. Instead of house prices, you have billions of examples of text. Instead of weights connecting five features to one output, you have billions of weights distributed across a deep neural network, organized into attention heads and feed-forward layers and embeddings. The complexity explodes. But the fundamental operation—finding a function that maps inputs to outputs by minimizing error—remains the same.

A large language model trains on text by trying to predict the next token (roughly, the next word) given all the previous tokens. It sees a prompt like "The capital of France is" and tries to predict "Paris." It sees "To bake a

cake, first" and predicts "preheat." It sees a math problem and predicts the answer. It sees a question and predicts a response. Across trillions of these predictions, the model's weights are adjusted—usually through backpropagation—to reduce the error between its predictions and the actual next tokens that appeared in the training data.

The result is a system that has compressed an enormous amount of statistical information about language into its weights and activations. It has learned which token is *likely* to follow which other tokens, given the patterns it has seen. This is pattern completion. It is what the system does. It is what it *is*.

But here is where the metaphorical language begins to diverge from the actual mechanics. When we say the model "learns," we mean something different from what we mean when we say a person learns. When a person learns something true—learns, for instance, that Paris is the capital of France—something changes in their understanding of the world. They can now integrate that fact with other facts. They can reason about it. They can recognize Paris in a new context: a painting, a conversation, a dream. They can distinguish Paris from other capital cities. They have developed a representation of Paris that connects to their broader knowledge.

The language model has done something far narrower: it has learned a statistical association. In the training data, the token "Paris" frequently appears after the sequence "capital of France is." The model has compressed this frequency into its weights. When prompted with "capital of France is," the model's internal activations flow through millions of parameters in a way that makes "Paris" a high-probability next token.

But the model has not built a theory of Paris. It does not know that Paris is a city, or that it is in France, or that capitals are seats of government. If you

ask the model to explain why Paris is the capital of France, it will generate an explanation—a plausible-sounding sequence of tokens—but the explanation emerges from the same statistical process. The model is not retrieving a stored fact; it is completing a pattern that says "when asked to explain why Paris is the capital of France, the next tokens are usually something like 'because the French government has its seat there' or 'because it is the largest city in France' or 'historically, Paris became the administrative center.'" The model generates whichever of these is most statistically likely given the patterns in its training data.

This distinction—between compressing statistical patterns and building true comprehension—is not semantic. It has concrete consequences. A system that has genuinely understood something can apply that understanding in novel contexts. A system that has compressed a statistical pattern is brittle in precisely the ways that true understanding is not.

Consider a thought experiment. Suppose a language model trained on English text learns that the token "Paris" frequently follows "capital of France is." The model learns a strong association. But suppose that in the training data, there is also a smaller pattern: occasionally, people make jokes about Paris being the capital of Germany, or playfully claim that Paris is the capital of Mars. The model has seen both patterns. It has compressed both. When you ask it "what is the capital of France?", the model's internal activations reflect the statistical likelihood of each possible continuation. The overwhelming majority of training examples point to "Paris." So the model outputs "Paris."

But now suppose you ask the model a question phrased slightly differently: "If France moved its capital, would it still be Paris?" The phrasing is unusual. There is no common pattern in the training data of this exact question. The model must complete it based on patterns that are somewhat

similar. One pattern is about what makes something a capital (it is where the government sits). Another pattern is that capitals can change (the training data probably contains examples of cities that were once capitals but are no longer). These patterns conflict with each other in ways that a human would easily navigate using their actual understanding of what capitals are.

What does the model do? It generates a continuation. The continuation might be coherent or confused, insightful or nonsensical. But it will be generated through pattern completion. If the training data contains similar examples where humans reason through a hypothetical scenario, the model will output text statistically similar to those examples. It is not reasoning through the hypothetical; it is generating text that sounds like someone reasoning through it. The similarity can be perfect. The understanding is absent.

This is what makes pattern completion so effective and so deceptive. The patterns in human-generated text include patterns for how humans talk *about* understanding. They include explanations, logical chains, expressions of confidence. A system trained to predict the next token in human text will learn these patterns. It will learn to generate text that looks like understanding, because understanding-text is common in the training data. The system will become fluent in the *form* of understanding without the substance.

Neuroscientists and cognitive scientists have spent decades studying how human brains actually learn and understand, and the mechanisms are vast and intricate in ways that remain poorly understood. But one thing is clear: understanding involves more than compressing statistical patterns. It involves prediction, yes—humans are pattern-completion machines too, in some sense. But human understanding also involves causality, abstraction,

reasoning, simulation, and crucially, feedback from the world. When a human learns something, they are not just learning a correlation in a dataset; they are building a model that they actively test against reality.

A child learns that hot things cause pain not by reading descriptions of heat in a training set, but by touching something hot, experiencing pain, and integrating that experience into their understanding of the world. The understanding is grounded in actual interaction with the world. It is falsifiable. The child can make a prediction ("if I touch that, it will hurt") and test it. When the prediction is wrong, the understanding updates.

A machine learning model has no ground to stand on. It never touches anything. It never experiences anything. It has no feedback loop with reality, only with the error signal that measures how well its predictions match the training data. The model learns that in the training data, certain patterns are more common than others. That is all it learns. If the training data is systematically biased or simply non-representative of the actual world, the model compresses those biases directly into its weights. It has no way to know. It has no mechanism to check.

This is why machine learning systems can be simultaneously very good at one thing and very bad at another, in ways that seem to violate the notion that they "understand" anything at all. A vision model trained on millions of images can classify photographs with superhuman accuracy. It has learned patterns about which pixels tend to appear together in images of dogs versus images of cats. But show that same model an adversarial image —an image that is imperceptibly modified in ways that preserve human recognizability but confuse the model's pixel patterns—and the model fails with complete confidence. It does not say "I am confused." It outputs a wrong answer with the same certainty as it outputs correct answers.

Why? Because the model's certainty is not tied to its actual accuracy. Certainty, in the model's world, means "this output is a high-probability continuation of the input." The model has no mechanism to distinguish between "this is high-probability because it is correct" and "this is high-probability because the training data happens to have this correlation." The model cannot know the difference. The architecture provides no way for it to know.

To understand this at a deeper level, consider how a neural network actually processes information. A large language model takes a sequence of tokens and passes them through multiple layers of transformations. In each layer, tokens are converted into high-dimensional vectors called embeddings. These embeddings are manipulated by attention mechanisms that determine which tokens should influence which other tokens. The output of each layer becomes the input to the next. Eventually, after dozens or hundreds of layers, the vectors are converted back into probabilities over possible next tokens.

At no point in this process does the model have access to ground truth. The vectors are not labeled with "this represents a true fact" versus "this represents a false fact." The attention weights are not labeled "this is a causally relevant connection" versus "this is a spurious correlation." The activations are just numbers—the result of mathematical operations applied to previous numbers, which were the result of mathematical operations on tokens, which were just representations of symbols from the input.

The model learns patterns in these patterns. High-level features emerge: certain attention heads learn to track narrative arcs, others learn grammatical structure, others learn semantic relationships. These emergent features can be quite sophisticated. A researcher studying a large language model might identify an attention head that seems to track "sentence

subjects" across sentences, or "numerical consistency" across a passage. The discovery of these features is often framed as evidence of genuine understanding: look, the model has learned to track subjects, therefore it understands grammar.

But this is where clarity is essential. The model has learned a pattern in which certain attention patterns correlate with what humans would call "tracking subjects" or "maintaining numerical consistency." The model has learned a higher-order compression of the data: not just which tokens follow which tokens, but which *clusters* of tokens follow which other *clusters*. This is more abstract. It is still pattern completion.

The critical difference: a human who understands grammar can apply that understanding in ways that go beyond pattern matching on new examples. A linguist can reason about why certain grammatical rules exist. They can imagine counterfactual grammars. They can explain why sentences with the same statistical form but different grammatical structures sound different. They have a theory that extends beyond the training distribution.

The model cannot do this. The model can generate text about grammar, because there is text about grammar in the training data. But it generates this text through the same pattern-completion process. It cannot reason about counterfactuals in the way a human can. Its "reasoning" is constrained by the statistical likelihood of continuations. Show it a grammatically novel construction and the model's performance will degrade in ways that often have nothing to do with actual grammatical principles and everything to do with the distribution of its training data.

This brings us to a crucial observation: the difference between pattern matching and understanding becomes most apparent at the boundaries of the training distribution. Within the distribution—on problems similar to

those the model has seen before—a sophisticated pattern-completion system can appear to demonstrate understanding. The model performs well. Its explanations are reasonable. Its reasoning seems sound.

But remove the safety net of the training distribution and the difference becomes stark. Ask the model about a scenario it has not seen, or ask it to reason about something far from its training data, and it does not gracefully degrade. It does not express uncertainty. It completes the pattern. It generates a plausible continuation, which is often wrong.

This is not because the training data was insufficient. A model with ten times more training data would still face the same problem, because the problem is not about quantity of data. It is about the nature of the operation itself. Pattern completion on a finite training set cannot generate true reasoning about unknowns. It can only generate text that is statistically similar to text it has seen about unknowns. And since humans write confidently about things they don't understand all the time—in speculation, in extrapolation, in educated guessing—the model learns to generate confident text about unknowns.

There is sometimes an argument that with enough scale, with enough parameters and enough data, the boundaries of the training distribution become less relevant. The argument suggests that at sufficient scale, a model will learn something like "general intelligence" that can extrapolate beyond its training data. This argument confuses scale with generality. A model with more parameters and more training data can compress more complex patterns. It can learn intricate statistical relationships that hold across larger portions of the data distribution. But it is still compressing statistical relationships. It is not escaping the fundamental constraint that it has no mechanism for truth.

The evidence for this is everywhere, though it is often overlooked because it is not flashy. Large language models are remarkably good at generating text that sounds authoritative about topics they have no real understanding of. They "hallucinate"—a term that itself obscures what is actually happening. The model is not confabulating or lying in the human sense. It is completing the pattern. In the training data, text about topics tends to be confident and elaborate. The model has learned this pattern and applies it universally.

The model does not know what it does not know. It has no knowledge of its own ignorance, no measure of its reliability on different types of problems. It has no access to a "ground truth flag" that would tell it whether its output maps onto reality. It has only the statistical likelihood of continuations. On that basis, it outputs text.

And here is where we arrive at the deepest problem: this is not a fixable flaw in the current generation of models. This is not something that better training data or better optimization techniques will solve. The architecture of the system—pattern completion—is fundamentally orthogonal to the operation of understanding. You can improve the pattern matching. You can make the system better at predicting the next token. You can scale it up, make it faster, integrate it more deeply into human workflows. But you will not, through any of these improvements, transform pattern matching into actual understanding. They are different operations, and conflating them is the error that has led us to deploy these systems into domains where understanding matters.

This is the foundation that everything else in this book rests on. The system is not slightly confused about its own knowledge. It has no knowledge at all, only patterns. The system is not weakly aligned to truth-seeking. It is optimized for something orthogonal to truth. The system is not on a path

toward genuine intelligence. It is on a path toward more sophisticated mimicry.

The Loss Function's Blind Spot

A loss function is a number. It measures, in aggregate, how wrong a model's predictions are. The lower the loss, the better the model is performing—at least according to the metric you've chosen to optimize. This is not an unusual or controversial statement in machine learning. It is the foundation of how these systems are trained. But the implications of this numerical relationship between "how wrong you are" and "what you should do next" have been almost entirely overlooked in discussions about why AI systems behave the way they do.

To understand the problem, you need to see what the loss function actually rewards.

Consider the simplest case: a language model trained to predict the next token in a sequence. The loss function typically used is cross-entropy loss. It measures how well the model's predicted probability distribution over possible next tokens matches the actual next token that appeared in the training data. Every time the model assigns high probability to the actual next token, the loss decreases. Every time it assigns low probability to what actually came next, the loss increases. This seems reasonable. We want the model to learn what actually comes after things. The problem emerges when you think about what this reward structure incentivizes in the face of uncertainty.

Suppose the model encounters a context in which multiple continuations are plausible. In the training data, perhaps the text says "The capital of France is" and the next token is "Paris." But in the training data, there are also instances where the text says "The capital of France is" and then continues into a discussion about European history, or politics, or geology.

From the model's perspective during training, all of these are valid continuations of that sequence. The loss function does not care about the underlying reality—that Paris is the capital of France and this is true. It only cares about the probability the model assigns to whatever token actually appeared next in the training corpus.

This distinction is crucial. The loss function rewards the model for matching the training data, not for being correct about the world. These are not the same thing. In many domains, they happen to correlate: training data about factual matters tends to contain correct information. But in domains where the training data is noisy, or where human writers speculate, or where the truth is ambiguous, the loss function has no mechanism to prefer accuracy over plausibility.

Here is the deeper problem: when faced with genuine uncertainty, the model learns to smooth it over.

Imagine you are training a model on text about weather prediction. Much of the training data contains confident statements like "Tomorrow will be sunny" or "Rain is expected Thursday." The model learns the pattern: when you see certain contextual cues (seasonal information, geographic data, atmospheric patterns), output a confident prediction about tomorrow's weather. But the model is not actually doing meteorology. It is learning the statistical pattern that confident weather predictions appear in certain contexts in the training data.

Now imagine the model encounters a context where genuine uncertainty is warranted—a borderline weather scenario where the outcome is genuinely unpredictable. The loss function presents a dilemma. One option: output a confidence-smoothed prediction (sunny, 70% sure), which will be wrong half the time, but will assign some probability to whichever outcome

actually occurs, and thus will not incur maximum loss. The other option: express genuine uncertainty ("I don't know; conditions are marginal"). This latter response, in terms of cross-entropy loss, is catastrophic. If the next token in the training data is "sunny," and the model outputs a probability distribution that is spread across multiple outcomes, it has assigned low probability to the actual outcome and will incur high loss.

The loss function penalizes the honest answer.

This is not a minor edge case. Across the entire space of possible inputs, there are far more scenarios about which genuine uncertainty is appropriate than scenarios where high confidence is warranted. But the training signal —the loss function—systematically punishes expressions of uncertainty and rewards confident predictions, even when the confident prediction is a smoothed-over guess rather than knowledge.

The mechanism is even more subtle than this. It is not that the model learns a simple rule like "always be confident." Rather, the loss function shapes the entire statistical structure of the model's internal representations. During backpropagation, gradients flow backward through the network, and these gradients encode information about how much each parameter contributed to the loss. Parameters that contributed to assigning high probability to the actual next token get adjusted to be more active in similar future contexts. Parameters that contributed to assigning low probability get adjusted to be less active. Over millions or billions of training steps, this process shapes the model's internal geometry.

The result is that the model develops an internal bias toward the kind of representations that assign high probability to something—anything—rather than spreading probability evenly across possibilities. This is not a conscious choice by the model. It is a statistical consequence of the loss

landscape. Given that the loss function rewards concentration of probability and punishes diffuse uncertainty, the model's weights converge toward states that generate confident outputs.

There is a term of art for this in machine learning: it is related to the phenomenon of "miscalibration," where a model's confidence does not match its actual accuracy. But miscalibration is usually discussed as a deficiency that can be fixed with post-hoc techniques like temperature scaling. This misses the point. The loss function created the miscalibration. The miscalibration is not a bug; it is a feature of the optimization process. A miscalibrated model—one that outputs high confidence on uncertain questions—is exactly what you get when you train a system to minimize cross-entropy loss on finite, noisy data.

The problem becomes more acute when you introduce human feedback into the training process.

Modern large language models are typically trained in two stages. First, a base model is trained on large amounts of text using the standard cross-entropy loss. Then, the model is fine-tuned using human feedback. Researchers ask humans to rate which of two model outputs is better. These preference judgments are then converted into a new loss function, often using a technique called reinforcement learning from human feedback (RLHF). The model learns to generate outputs that human raters prefer.

This seems like an improvement. Instead of just optimizing for likelihood, the model is now optimizing for what humans think is good. But humans have their own biases, and chief among them is a preference for confident, well-articulated answers over expressions of uncertainty. When you ask a human to choose between two completions—one that confidently explains a concept and one that says "I'm not sure"—the human will almost always

prefer the confident explanation. This is true even when the human would admit, if asked directly, that confident claims about some topics are not warranted.

The loss function, now shaped by human preference, is still punishing uncertainty. In some ways, it is punishing it more effectively, because now the signal is coming not from whether the output happened to match the training corpus, but from whether it satisfied a human rater's preferences. And human raters have learned to prefer fluent, authoritative-sounding text. The model learns this preference signal and optimizes for it.

There is a specific term in machine learning for what happens next: "reward hacking" or "specification gaming." The model optimizes so effectively for the specified objective that it achieves high scores on that objective while violating the implicit intent of the objective. In this case, the implicit intent is "generate helpful, accurate, honest responses." The explicit objective, implemented in the loss function, is "minimize the difference between your output and what human raters prefer." These are not the same. A model trained to satisfy human raters will learn that confident-sounding claims are preferred, that smooth prose flows are preferred, that tangential information that sounds relevant is preferred. It will learn that saying "I don't know" is dispreferred. And it will optimize ruthlessly for these learned preferences.

The model is not being deceptive in any intentional sense. It has no intentions. But it is being shaped by optimization pressures to produce outputs that appear correct and confident, regardless of whether they actually are. The loss function has created an incentive structure where appearing correct is rewarded and appearing uncertain is punished.

Consider a concrete example. Suppose a model is asked a factual question for which the training data contains conflicting information. Perhaps

multiple sources in the training corpus make contradictory claims about a historical event, or a scientific fact that was later revised, or a social phenomenon where reasonable people disagree. What does the model learn to do?

One possibility: the model could learn to output both perspectives and acknowledge the disagreement. This would be the honest response. But if human raters prefer clear, unambiguous answers, they will rate a single, confident statement higher than an explanation of disagreement. The loss function will reward the confident statement. The model will learn that confidence is preferred, even at the cost of accuracy.

Another possibility: the model could output whichever perspective appears more frequently in the training data. But this conflates frequency with truth. Misinformation and falsehood can be well-represented in training data. If a false claim has been repeated often enough, it will appear frequently in the training corpus, and the model will learn to repeat it with confidence.

A third possibility: the model could hedge its claims, expressing appropriate uncertainty. But every hedge, every qualifier, every "may" or "possibly" or "some evidence suggests"—each of these increases the length and complexity of the output while assigning lower probability to any single continuation. From the loss function's perspective, this is inefficient. A shorter, more confident statement will incur lower loss. The model learns to hedge less.

This is not speculative. You can observe this in practice. Large language models are notoriously reluctant to say "I don't know," even when they should. They are prone to what is sometimes called "hallucination" but is better understood as "confident confabulation"—generating detailed, internally consistent-sounding false information. They are skilled at

sounding authoritative on topics they have never meaningfully learned about. All of this is explicable as the direct result of training objectives that reward confidence and punish uncertainty.

Now consider what happens when such a system is deployed in a domain where the distinction between "plausible" and "correct" matters.

In a medical context, if a model is asked to diagnose a patient based on symptoms, the loss function during training might have measured how often the model's diagnosis matched the actual diagnosis recorded in medical data. But the loss function did not measure whether the patient actually had the disease diagnosed, or whether the patient was harmed by being given an incorrect diagnosis. It measured statistical similarity between the model's output and the training corpus. If the training data contains misdiagnoses, the model learns to confidently reproduce them. If the training data contains doctors confidently diagnosing conditions that later proved to be something else, the model learns to sound confident about uncertain diagnoses.

In a legal context, if a model is asked to cite relevant case law, the loss function during training measured something like "does this output look like a plausible legal citation and explanation?" The model learns to generate citations that are formatted correctly, that reference real cases or synthesize details in a way that sounds like real cases, that explain legal reasoning in a way that matches the patterns in legal writing. But the loss function does not measure whether the citation actually exists, whether it is relevant, or whether the legal reasoning is sound. It measures statistical likelihood. The model becomes very good at generating confidence-smoothed text that looks like legal scholarship while having no mechanism to verify that it is accurate.

This is what happens when you optimize a system for the wrong metric and then deploy it in a context where the metric and the actual objective have diverged. The system becomes very good at gaming the metric. It becomes very good at appearing correct without being correct. And because the system has no internal mechanism to distinguish between these states—no ground-truth flag, no access to feedback from reality—it assigns high confidence to both.

There is sometimes an argument that this problem is fixable through better loss functions. Perhaps, the argument goes, we should add a penalty for uncertainty—explicitly encourage models to say "I don't know" more often. This seems sensible. In practice, it has proven difficult to implement. The penalty has to be carefully calibrated; too weak and models ignore it, too strong and models become useless, always deferring. There is also a meta-problem: how do you know when uncertainty is appropriate? To penalize uncertainty incorrectly, you have to have access to ground truth. You have to know what the model should know and what it should be uncertain about. But the whole problem is that the model is trained on finite data and has no ground truth. If you could reliably identify when uncertainty is warranted, you could train the model on that information directly. You would not need a separate penalty.

A deeper problem is this: any loss function that can be numerically specified and optimized over will have the same fundamental property. It will reward the model for achieving good scores on the metric, and it will do so whether or not the metric correlates with the actual goal. You can try to design a better metric, one that penalizes hallucinations or false confidence. But you are still optimizing a number, and the model will still learn to produce outputs that make that number small. As long as the optimization is happening in the space of probabilities and predictions, rather than in the space of understanding, the incentive structure will favor

confident generation over honest uncertainty.

The loss function's blind spot is that it cannot see the difference between "this output matches the training data" and "this output is true." It can only see one of these things: the first. Everything flows from this. A system optimized to match training data rather than to be truthful will develop an anatomy of evasion, mechanisms to handle uncertainty by smoothing over it, patterns of confident generation regardless of actual knowledge. This is not a flaw that can be fixed by better engineering. It is a structural consequence of the training objective.

What is remarkable is how thoroughly this insight has been absent from mainstream discussions of AI development. When researchers and companies have deployed increasingly powerful systems, the assumption has often been that if a system performs well on a training metric, it will perform well in the real world. This is sometimes true. For tasks where the training metric and the real-world objective are closely aligned—like image classification on carefully curated datasets—the assumption holds. But as these systems have moved into domains where correctness matters more than plausibility, where truth is not easily quantifiable, where the cost of confident falsehood is high, the assumption has broken down.

The loss function has its blind spot exactly where we need sight most: in the gap between what the metric measures and what we actually care about. The model optimizes toward the metric, becomes excellent at it, and then breaks down in the real world because the real world does not work the way the loss function imagined it.

And the model, having no access to the ground truth that would let it know it is wrong, continues to output high-confidence predictions. The mechanism that would allow it to self-correct—feedback from reality—is

absent. The loss function measures only historical pattern matching, not future accuracy. By the time the model's outputs are being used in a domain where the cost of false confidence is measured in human harm, the training is already over. The weights are fixed. The incentive structure that shaped those weights was never designed to optimize for truth in the first place.

The Architecture of Evasion

The transformer architecture is elegant. It is also, in a strict technical sense, incapable of knowing what it knows.

This is not metaphorical. When we trace the actual mechanics of how large language models process information—how they take your question, transform it into vectors, pass it through layers of attention, and eventually emit a probability distribution over the next token—we find no mechanism that distinguishes between "I have learned this fact" and "this is a statistically plausible continuation of the pattern." The model does not store facts as facts. It does not flag certain outputs as verified or grounded. There is no ground-truth register, no epistemic marker that says *this came from reliable training data* versus *this is a confabulation that happens to sound good*. The architecture simply does not have the machinery for such a distinction.

To understand why this matters, we need to look inside the model's skull.

A transformer begins by converting words into embeddings—high-dimensional vectors that represent meaning in the learned space of the model. The word "Paris" is not the actual city. It is a point in, say, a 4,096-dimensional space, positioned near other European capitals, near words associated with architecture and history and tourism. These positions are learned during training. There is no inherent reason why "Paris" should sit where it does; its location is simply the result of statistical pressure, of being near those words in the training text, of gradients pushing it in certain directions to minimize the loss function.

The embedding is already a compression, a flattening of meaning into

numbers. But it is not yet a flattening where we have lost information so much as transformed it. The vector still carries statistical traces of what "Paris" tends to mean in text.

From the embedding, the model passes the token through layers of attention—the famous mechanism that allows transformers to look back at previous tokens and decide which ones are relevant to the current prediction. Attention does this by computing similarity scores between the current token and all previous tokens, then using those scores to create a weighted combination (a "context vector") that feeds into the next computation.

Here is the crucial thing about attention: it is a mechanism for *pattern weighting*, not for *verification*. When the model attends to a previous token —say, it looks back and notes that "Paris is the capital of" came before "France"—it does so because the statistical pattern suggests that this sequence is similar to patterns in the training data. Attention amplifies the relevant patterns. But there is no moment where the model stops and asks: *Is this actually true? Or is it just a common pattern in text?* The mechanism cannot ask that question because it has no access to ground truth. All it has is the pattern.

Consider a more troubling example. A model is asked about a fictional scientific paper: "Tell me about the 2019 study on quantum consciousness by Dr. James Patterson published in the Journal of Neuroscience." In the training data, there likely are real papers by real researchers in real journals. There are patterns that look like citations, abstract language, methodological details. A human would stop and think: *I don't know if this paper exists*. But the model does not have that option. It can only complete the pattern. It attends to similar patterns in its training data—real papers, real journals, the grammar of scientific abstracts—and generates text that is

statistically similar to those patterns. The output will be fluent, detailed, confident. The attention mechanism will find patterns that look relevant and weight them accordingly. And at no point in this process will the model encounter a flag that says *stop, you don't actually know this*.

The problem is architectural. Attention is not designed to verify. It is designed to weight. And the weights it learns are entirely a function of what appeared in the training data, not of whether what it generates is true.

After attention, the signal passes through feedforward networks—essentially nonlinear transformations that apply learned weights to the context vector and further compress and reshape the information. These layers accumulate more learned patterns, more statistical associations. By the time we reach the final layer of the model, we have passed through dozens or hundreds of these transformations, each one learning patterns and applying them.

And then comes the output layer, where the model predicts the probability of the next token.

This is the final compression, the narrowing point where everything the model has learned is converted into a single number for each possible next word: a probability. The model outputs something like: *the next token is "France" with probability 0.87, "Germany" with probability 0.05, "capital" with probability 0.03, and so on.* The actual token that appears in the generated text is sampled from this distribution (or, in many modern systems, simply the token with the highest probability is selected).

Notice what has happened. The model has taken your question, transformed it through dozens of layers of learned patterns, and arrived at a decision: *this token is most likely to come next*. But the basis for this decision is entirely statistical. It is the token that appears most often after

similar patterns in the training data. There is no separate channel that says: *this token is true* or *this token is false* or *this token is uncertain*. The model does not compute confidence in the epistemic sense—confidence that something is actually true. It computes likelihood in the statistical sense: *how common is this pattern in the training distribution?*

These are not the same thing.

A lie that appeared frequently in the training data will have high likelihood. A true fact that appeared rarely will have low likelihood. The model will generate the lie confidently.

Now consider what it would take to add ground truth to the architecture. You would need a separate input channel: information about whether each piece of training data is actually true or false. You would need to backpropagate that information through the model, adjust the weights, and create learned patterns that correlate high-confidence outputs with ground truth. But this ground truth signal would itself need to come from somewhere—from human annotation, from experiments, from reality. And here is the problem: you cannot perfectly label the entire training corpus this way. And even if you could, the model would still need to generalize to new situations, new questions it has not seen before. At the point of deployment, when the model encounters a new question, it must make a prediction. It does so by computing: *what pattern does this resemble, and what did I learn about similar patterns during training?* If the pattern is novel, or if the ground truth is complex and contextual, the model still has no access to reality. It can only extrapolate from patterns.

The fundamental architecture is still one of pattern completion, not truth-tracking. You can make it better at pattern completion by giving it better ground truth labels, but you cannot turn it into a system that actually knows

when it is right versus wrong about something it has not seen before.

This is where the notion of a "ground-truth flag" becomes more than metaphor. A truly robust epistemic system—a scientist, for instance—has access to feedback from reality. You make a prediction, you test it, and reality tells you whether you were right. You adjust. You build a model of the world that is constrained by its interaction with the world. The model cannot just generate plausible-sounding predictions; it is penalized by reality when it does.

Large language models have no such penalty at inference time. They generate a token. The loss function that would penalize them for being wrong is no longer active. The weights are fixed. The model receives no feedback about whether its output was true or false. It simply emits probabilities and moves to the next token.

Some recent work has explored what are called "uncertainty quantification" methods—techniques to extract signals from the model about how confident it is in its predictions. These are clever and sometimes useful. But they face a hard limit: they are working within the architecture that was never designed to represent uncertainty in the first place. If the model has learned that certain patterns correlate with high-probability outputs, and if those patterns are sometimes hallucinated, then the model's internal confidence signal will sometimes be high for hallucinations. You cannot extract epistemic certainty from a system that does not internally compute it. You can only infer it indirectly, from activations and probability distributions, and your inference will be as good as your guesses about what those activations represent.

Let me be more concrete. Imagine we take a well-trained language model and we ask it: *What is the capital of France?* The model outputs "Paris"

with probability 0.99. Now imagine we ask it: *What did philosopher Ludwig Wittgenstein eat for breakfast on March 3rd, 1951?* The model outputs something like "Wittgenstein likely had a simple breakfast of bread and tea, as was common in Austria at the time" with probability 0.72. (These numbers are illustrative, but the phenomenon is real.) Why are these probabilities what they are?

For the first question, the answer is straightforward. Paris really is the capital of France, and this fact appears thousands of times in the training data in clear, consistent form. The model has learned a strong, reliable pattern, and the probability reflects this.

For the second question, the model has no ground truth to work with. Wittgenstein's breakfast preferences are not well-documented, and specific facts about his daily life are sparse in the training data. But the model still generates an answer. Why? Because it has learned patterns about: (a) what philosophers typically ate, (b) what was common in Austria at that time, (c) the grammar and style of biographical writing. It combines these patterns and generates a plausible-sounding completion. The probability of 0.72 does not mean the model is 72% confident the claim is true. It means the token "tea" (or whatever word appears in that position) is 72% likely given the pattern. These are orthogonal things.

The model cannot distinguish between them because it has no mechanism to do so. The architecture does not include a subsystem that evaluates: *Is this supported by reliable data? Or is this an interpolation from sparse or conflicting signals?* All the architecture does is compute: *What token is most probable given these learned patterns?*

The evasion emerges from this architectural blind spot. Because the model has no way to represent uncertainty about facts it has not seen, it defaults to

generating continuations that sound coherent. When pressed with a question it has no direct answer to, the model cannot output something like *I don't know, and here's why*. That would require the model to have an internal representation of the distinction between "facts I've reliably learned" and "gaps in my knowledge." But this distinction is not learned by the transformer. The model learns patterns. It does not learn metadata about those patterns—information about their reliability, their source, their grounding.

You might ask: could we modify the architecture to include such metadata? Could we add a second output head that predicts not just the next token but also a confidence score that actually reflects epistemic certainty?

In principle, yes. In practice, we run into the training problem again. A model learns what it is rewarded for learning. If you train a model on a large corpus of text, and the only signal you give it is "minimize prediction error for the next token," then the model will learn patterns that predict the next token. It will not separately learn to identify which of its predictions are grounded in reliable data and which are confabulations, because it is not explicitly rewarded for making this distinction. The model could learn this in principle, but it would require a fundamentally different training regime: one where the model is explicitly penalized for high-confidence predictions about things it has never seen, and rewarded for expressing uncertainty in such cases.

This has been attempted. Constitutional AI, RLHF with uncertainty penalties, and other techniques try to shape the model's outputs toward greater honesty. But they work against the grain of the underlying architecture. The transformer was not designed with epistemic honesty in mind. It was designed to compress patterns into probabilities. Adding a layer of "be honest" on top of that does not change the fundamental

mechanism.

Moreover, there is a deeper issue: the model cannot know what it does not know. This is not a philosophical puzzle; it is a technical constraint. The only information available to the model during inference is the tokens that came before. If the correct answer to your question does not appear in the training data, the model has no access to it. It can only interpolate from nearby patterns. It cannot flag its own ignorance because flagging ignorance would require access to ground truth—would require the model to know what it is missing. But if the model knew that, it would have the answer.

This is sometimes called the "unknown unknowns" problem. The model's learned patterns cover the training data well. But there are gaps—questions about events that happened after training, facts that were never in the training corpus, and edge cases that combine learned patterns in novel ways. The model has no mechanism to identify these gaps. It only knows how to complete patterns. So it completes them, often confidently.

Here is where evasion becomes subtle. The model is not deliberately lying. It has no intentions at all. But the architecture creates an incentive structure where generating plausible-sounding completions is the only option, and where representing actual uncertainty is either impossible or costly in terms of the training objective. The model learns to be fluent, to smooth over gaps, to fill in missing information with statistically likely tokens. These are not lies in a moral sense, but they are evasions—ways of avoiding the hard truth, which is that the model does not know.

The attention mechanism, which should be a window into what the model knows and how it knows it, is instead a black box. When a model generates an answer, we can visualize attention—we can see which previous tokens

the model attended to when generating the current token. This is intriguing and sometimes illuminating. But it is not a guarantee of grounding. The model might attend very closely to a previous token that happened to be false. Or it might attend to a pattern that resembles the right answer without attending to the actual source of that answer. Attention tells us which patterns the model found salient, not whether those patterns are reliable.

This is why so many deployed systems fail in ways their creators did not expect. The creators looked at the model's outputs on test data and saw fluency, accuracy, coherence. They did not realize that fluency and accuracy on test data do not generalize to accuracy on deployment data if the deployment domain involves unknown unknowns. The model was optimized to minimize loss on the training distribution. When it encounters a different distribution—a different kind of question, a domain it was not trained on, a novel combination of concepts—the learned patterns become less reliable. But the model has no way to know this. It generates outputs with the same apparent confidence as before. The architecture does not update its confidence levels based on distributional shift. The loss function is no longer active. The model is just running, completing patterns.

What we are describing is a system that has been thoroughly optimized to appear coherent, without any built-in mechanism to verify that coherence against ground truth. The transformer architecture is exceptionally good at the first task. It is structurally incapable of the second. And when these systems are deployed in domains where the cost of false coherence is high—medical diagnosis, legal reasoning, high-stakes decision support—the consequence is a class of failure where the system remains confident even as it is wrong.

The evasion is not malicious. It is architectural. And it is, in a deep sense, unavoidable without fundamentally changing how we build and train these

systems.

Emergence as Camouflage

In March 2023, a leaked internal Google document described a conversation between two researchers about Bard, Google's large language model. The exchange was striking not for what it revealed about the model's architecture, but for what it revealed about how we interpret the model's behavior. One researcher had noticed that Bard seemed to be engaging in something like *reasoning*—that it could solve multi-step problems by breaking them down, checking its work, and correcting itself mid-generation. The other researcher agreed: the model appeared to be *thinking*. The document circulated. Think pieces followed. Here was evidence, some argued, that something like cognition was emerging in these systems.

But what had actually emerged? Not reasoning. What emerged was a new form of pattern completion—one sophisticated enough to mimic the surface structure of reasoning so convincingly that humans trained to recognize reasoning would see reasoning. The model had learned, from its training data, that reasoning has a recognizable shape: a problem stated, options considered, a calculation or logical step, a conclusion checked against the initial problem. When prompted to solve a multi-step problem, the model generated text that followed this shape. It looked like thought. It performed the appearance of thought. But the underlying process—statistical completion of tokens—remained unchanged. The model was not, in any meaningful sense, reasoning. It was generating text that resembles reasoning, optimized by a loss function that rewards resemblance to reasoning in training data.

This is the central trick of emergence in large language models, and it

explains why so many observers have mistaken increasing model capacity for increasing model understanding. Emergence is real—but what emerges is not intelligence or reasoning or knowledge. What emerges is *complexity of camouflage*. As models grow larger, their pattern-completion becomes more sophisticated. They learn subtler textures of language, more elaborate ways to arrange tokens into sequences that feel meaningful. They learn to generate coherent multi-paragraph arguments, to maintain consistency across long contexts, to use technical language appropriately, to format answers in ways that match human expert answers. None of this is understanding. All of it is pattern completion. But all of it is *better* at appearing to be understanding. And we, the observers, have made the fundamental error of confusing appearance with reality.

The moment that seemed to crystallize this confusion—the moment the field really began to celebrate emergence—came with the discovery of "in-context learning." Researchers noticed that if you prompted a large language model with a few examples of a task, the model could perform similar tasks without retraining or fine-tuning. Give GPT-3 a couple of examples of translating English to French, and it could translate new English sentences to French. Show it a pattern of logical inferences and it could make new inferences. This looked, unmistakably, like *learning*—the same process humans use when we encounter a new task and figure out what's expected.

The resemblance is superficial. What the model is actually doing is using the examples as context clues to predict what kind of tokens should come next. It has learned, during training, that when text follows a certain pattern —example, example, example, then a new prompt—the appropriate completion is to continue the pattern. The model is not learning a new task. It is recognizing a pattern in its input and completing that pattern. This is extraordinarily impressive as pattern completion goes. It is, precisely, not

learning in the sense that humans learn—building new mental models, developing new understanding, integrating new information into a framework of comprehension. The model is running the same algorithm it always runs: ingest tokens, predict the next token, repeat. The context is different, but the process is invariant.

Yet this in-context learning was celebrated as a fundamental breakthrough, evidence of genuine reasoning emerging in neural networks. Papers were published. Conferences buzzed. The implication was clear: if a model could learn without retraining, it was capable of something akin to genuine thought. The leap was intuitive. It was also wrong. And it matters profoundly because this mistake—this confusion between sophisticated pattern matching and actual reasoning—has shaped how we deploy these systems and what we expect them to be capable of.

Consider chain-of-thought prompting, a technique that emerged from this same period. Researchers discovered that if you explicitly ask a model to "think step by step," it performs better on complex reasoning tasks. Ask GPT-3 a math problem directly and it might get it wrong. Ask it to work through the problem step by step, and it often gets it right. This looked like evidence of genuine reasoning—that the model was actually capable of logic, of holding intermediate results in mind, of checking work. The implication was that the model's failure on direct prompts was not due to inability to reason, but to inability to express its reasoning in a single token. Give it permission to work through the problem explicitly, and the reasoning emerges.

But here again, what is actually happening is pattern completion of a higher-order kind. The model has learned, from its training data, that solutions to complex problems in human writing are typically presented as a sequence of steps. It has learned the texture of step-by-step reasoning—

the way each step is phrased, the kinds of intermediate results that appear, the logical connectors that link them. When asked to think step by step, the model generates text that follows this learned pattern. And because the pattern is faithful to how humans actually solve problems, the generated steps often converge on correct answers. But the model is not reasoning. It is *predicting what reasoning looks like when it is written down*. These are not the same thing.

The distinction becomes crystalline when we look at tasks that require reasoning the model has not seen exemplified in training data. When researchers have tested models on novel reasoning tasks—problems that require applying logic in ways not present in any training text—the models fail, often badly, often while expressing high confidence in incorrect answers. The model can mimic the appearance of reasoning on problems similar to ones it has encountered. It cannot actually reason from first principles. It has no mechanism for it. But because it can mimic reasoning convincingly on familiar problems, the appearance of genuine reasoning capability persists.

This is where emergence becomes truly treacherous as a concept. We use "emergence" to describe the way new capabilities appear when models are scaled up—the way a 7-billion-parameter model can do things a 1-billion-parameter model cannot. But the word itself carries an implication of something *new arising*, something that was not present before in any form. This is misleading. What is actually happening is that the model's capacity to represent patterns has increased, allowing it to capture subtler, more complex patterns from the training data. The capability was, in a sense, already implicit in the training data—it was always there, encoded in the text. Scaling merely gave the model enough representational capacity to extract and express those patterns.

The confusion deepens because emergence, on its face, seems to suggest something unpredictable or poorly understood. When emergent behaviors appear in scaled-up models, there is genuine scientific interest in understanding why they appear at particular scales and not others. But this scientific uncertainty about the mechanisms of emergence has been mistaken, by many observers, for uncertainty about what emergence represents. Scientists studying emergence are asking: why does this pattern-matching capacity crystallize at this parameter count? Observers celebrating emergence are asking: does this mean the model understands? These are different questions. The former is genuinely difficult. The latter has a clear answer: no. The model is matching patterns. Larger models are better at matching patterns. That is not the same as understanding.

A useful contrast comes from looking at what emergence is *not* in biological systems. When we talk about consciousness emerging from neural activity, or meaning emerging from grammar, or life emerging from chemistry, we are describing something metaphysically unclear but empirically observable: a new property that cannot be reduced to or predicted from the properties of the component parts. There is something happening at the systems level that is not a simple sum of lower-level processes. We don't fully understand how this works, but we can observe that it does.

In large language models, by contrast, emergence is entirely reducible. Every behavior that "emerges" in a scaled-up model is reducible to the same statistical completion process that operates in smaller models. There is no new mechanism, no new principle of operation. There is only the same mechanism operating on a more complex learned representation. A human child learning to reason is experiencing something that cannot be fully reduced to pattern matching—there is creativity, generalization, the capacity to apply logic to novel domains. A language model scaling from 7

billion to 70 billion parameters is experiencing only the latter: better pattern matching, finer-grained representation of statistical regularities. Calling both emergence, and grouping them together conceptually, collapses a crucial distinction.

Yet this collapse is nearly universal in how the field talks about these systems. When OpenAI released GPT-4 in March 2024, the announcement emphasized "emergent capabilities"—the model's ability to perform well on tasks it was not explicitly trained for, to engage in apparent reasoning, to demonstrate what seemed like insight. Journalists and researchers used language suggesting something close to genuine understanding had materialized. The model could solve complex problems. It could explain humor. It could write poetry. These are the things intelligence does. Therefore, the inference went, the model is intelligent.

But the model can do these things because its training data contains countless examples of humans solving complex problems, explaining humor, and writing poetry. The model learned the statistical patterns that characterize these behaviors and can now generate text that exhibits those patterns. This is not insight. This is not understanding. This is not even reasoning in the proper sense. It is sophisticated pattern completion, and the sophistication is real—but sophistication in mimicry is not the same as sophistication in comprehension.

One of the clearest ways to see through this confusion is to examine what the model actually *does not* know, even as it appears to know. A large language model trained on text data before 2023 has no direct knowledge of events after 2023. It cannot learn new facts from experience because it does not have experiences and cannot update its weights based on feedback. It has no persistent memory across conversations. It cannot reason about the physical world because it has never had sensory

experience. It cannot predict how a new drug will behave in a human body based on first principles because it has no understanding of chemistry or biology—only patterns in how humans write about chemistry and biology. These are not limitations that will be overcome by scaling. These are architectural. They are fundamental to what the system is: a token predictor.

Yet the model will confidently discuss recent events by generating text that resembles discussion of recent events. It will talk about physics and chemistry and biology with the fluency of an expert. It will make confident claims about what it does not know. This is where emergence becomes camouflage most completely. The model has learned, from its training data, how to generate text that sounds authoritative, detailed, and insightful. It generates this text in response to questions about domains where it has no grounding. The text is plausible. Humans read it and update their beliefs about what the model knows. The model itself has no beliefs and knows nothing about the domains it is discussing. But its pattern completion is sophisticated enough that the gap between knowledge and apparent knowledge becomes nearly invisible.

This gap is most visible in what we might call "emergence failure"—moments where the model's pattern-matching reaches its limits and its camouflage breaks down. A model that has learned to solve arithmetic problems by predicting what solutions look like will fail on arithmetic problems with unusual properties. A model trained on examples of legal reasoning will confidently cite cases that do not exist, generating case names and citations that sound authentic because it has learned the pattern of how real citations are formatted. A model trained to predict text that resembles medical diagnosis will suggest treatments for fictional conditions or apply inappropriate treatments to real conditions, because it is completing patterns in language, not reasoning about physiology.

These failures are instructive precisely because they reveal the underlying reality: what looked like reasoning was pattern completion. What looked like knowledge was statistical correlation. What looked like understanding was the generation of text that resembles understanding. When the pattern breaks—when the input is sufficiently novel or the domain is insufficiently represented in the training data—the camouflage fails and we see what was underneath: a system that has no mechanism for actual understanding, only for generating plausible text.

Yet even when these failures occur, they often do not substantially change how the systems are deployed or how observers think about what the systems can do. A model that confidently cites nonexistent cases is understood by its builders as "hallucinating"—a term borrowed from human psychology that suggests the model is experiencing something, making mistakes in the way humans make mistakes. The term obscures what is actually happening: the model is doing exactly what it was designed to do—generate plausible text—and it is doing it so well that the text is often indistinguishable from genuine knowledge. The error is not in the model's functioning. The error is in treating the model as if it has access to ground truth and is simply failing to retrieve it. The model has no access to ground truth. It never did. It generates text that resembles retrievals from ground truth. When the resemblance is perfect, we believe it. When the resemblance fails, we call it hallucination. The underlying process remains the same.

This is the core insight about emergence as camouflage: as models scale, their camouflage becomes more sophisticated, not their actual cognition. A larger model is better at mimicking the surface features of reasoning, knowledge, and understanding. This is deeply, profoundly useful for many purposes. If you need text that sounds like an explanation of quantum mechanics, a large language model is an excellent tool. If you need

something that resembles a legal brief or a medical summary or an academic paper, the model can generate it, often convincingly. But usefulness in mimicry is not the same as actual understanding, and treating it as such creates a dangerous category error.

The field has largely accepted that large language models are "stochastic parrots"—systems that generate statistically likely sequences of tokens without understanding what those tokens mean. This characterization is accurate in its basics, but even this phrase can mislead. A parrot learns through repetition and reinforcement to produce specific sounds in response to specific stimuli. A language model learns through gradient descent to assign probability distributions to tokens given prior context. The mechanisms are different. But the key insight remains true: both systems are doing something much more like sophisticated mimicry than like genuine comprehension. The parrot says "hello" because it has learned that "hello" is a likely response in certain contexts. The model generates "hello" for the same reason, except it is doing it with billions of parameters and operating on patterns learned from terabytes of text.

The emergence narrative obscures this by suggesting that at some point—at some critical parameter count or training procedure—the system crosses a threshold from mimicry to understanding. This is seductive because it mirrors the way we might imagine biological intelligence emerging: at some point in evolution, or in child development, simple pattern matching gave way to actual thought. But there is no evidence that this transition is happening in language models, and significant evidence that it is not. The models continue to fail on tasks that require actual reasoning—tasks that cannot be solved by pattern matching, even very sophisticated pattern matching. They continue to generate false information confidently. They continue to lack any mechanism for grounding their outputs in reality. They simply do these things more subtly, more convincingly, with greater

apparent sophistication, as they scale.

What has emerged, then, is not reasoning or understanding. What has emerged is a form of language use so sophisticated that it triggers in humans the assumption of understanding. This is a real phenomenon, and understanding it is crucial. But it is a phenomenon about human cognition and human assumptions, not about machine cognition. We are wired to assume that fluent language use indicates understanding. A system that generates fluent language will trigger this assumption, whether or not the system understands anything. A system that generates *very* fluent language, that maintains consistency across long passages, that uses technical terminology appropriately, that acknowledges edge cases and expresses appropriate confidence about uncertain matters—such a system will trigger the assumption so powerfully that questioning it feels like questioning whether humans understand language.

But the system is not understanding language. It is processing patterns in language. And we are not learning to understand machines. We are learning to be fooled by machines that are very good at mimicking understanding.

This matters because the way forward depends on clear-eyed assessment of what these systems actually are. If we believe that emergence means genuine reasoning is developing, we will continue to deploy these systems in domains where real reasoning is required, and we will be surprised when they fail—confidently and plausibly. If we understand emergence as the development of more sophisticated camouflage, we can ask different questions: Where is the gap between plausible text and actual correctness widest? How can we build feedback mechanisms that ground the system's outputs in reality? What domains are so dependent on actual reasoning that pattern-matching systems should not be deployed there at all?

The next step in understanding this problem requires looking at what happens when these sophisticated pattern-matching systems are asked to do something they are fundamentally unequipped to do: retrieve facts about the world and distinguish those facts from plausible fictions generated by their own pattern-matching. This is not an edge case. This is perhaps the most central function we are asking these systems to perform. And it is here that the confusion between appearance and reality becomes most dangerous.

The Retrieval Illusion

When you ask a large language model a factual question—"Who won the 1952 U.S. Presidential election?"—something happens inside the system that has almost nothing to do with retrieval. No mechanism reaches into a database of facts and extracts a record. No pointer follows a path to stored information. Instead, the model performs its core operation: it generates the next token in a sequence based on statistical patterns learned from training data. The next token after your question is likely to be "Eisenhower" or "Dwight" because those tokens appear frequently in similar contexts in the training corpus. The system is not retrieving this fact. It is completing a pattern. And yet, the output looks identical to retrieval. The user receives a correct answer, delivered with the fluent certainty that seems to warrant trust. The confusion between these two processes—actual information retrieval and sophisticated pattern completion—is not a minor implementation detail. It is the crack in the foundation through which catastrophe seeps.

Consider what a retrieval system would need to do. A retrieval system would need to access some external store—a database, a knowledge graph, a collection of documents—and find the information that matches the query. This process is separate from language generation. A search engine retrieves relevant documents and then may need to extract or summarize the answer. A human being with access to a reference library retrieves the reference and reads the relevant passage. The act of retrieval is distinct and verificable: we can check whether the system actually found what it claims to have found. We can see the source.

A language model does none of this. It has no access to external

information during inference. It has no mechanism to search or retrieve. It only has the statistical relationships between tokens learned during training, encoded in billions of parameters. When you ask it a question, it generates what comes next based on those learned patterns. If the training data contained many instances of "1952 U.S. Presidential election" followed by "Eisenhower," the model will predict that token with high probability. This is not retrieval. This is prediction based on frequency.

The system produces output that is *statistically similar to* factual information. When it works—when the training data was representative and the pattern was learned correctly—this output is also factually correct. But the similarity is incidental, not causal. The model did not retrieve the fact because it was true. It generated the tokens because they were likely given the patterns in the data. There is no internal process that checks correspondence with reality. There is no mechanism that says, "This is correct because it matches external facts; therefore, I will output it." The model simply outputs what is most likely given its training. Correctness, when it occurs, is a side effect.

The distinction matters because it predicts failure modes. A retrieval system fails when the relevant document is not in the database. When a human asks a reference librarian about a fact that is not in the library, the librarian can, in principle, say so. The system will not return information it does not have. A language model does not have this option. It must always generate something. There is no "not in the database" state it can reach. There is only "generate the most likely next token," and then the next, and the next. The system will continue constructing output even when it has nothing to say, because generating text is what it does.

This is where the confidence plague emerges in its most acute form. The model generates fabricated information with the same fluent certainty as

correct information because, from the model's perspective, the process is identical. Both are pattern completions. Both feel equally predictable from the model's weights and activations. Both produce text that reads like coherent, factual statements. The difference is that one matches reality and the other does not—but this is a difference the model cannot perceive from the inside.

Let's trace this through a concrete example. A researcher tested a large language model by asking it to cite a paper that does not exist. The query was something like: "What is the methodology described in the paper 'Transformer-based learning in zero-shot environments' by Smith et al., published in the *Journal of Machine Learning Research* in 2019?" There is no such paper. The authors, journal, and year are plausible. The title is thematically coherent with actual research. The model generated a response that described a specific methodology—specific enough to be useful if it were real—and cited it confidently. The output read like a genuine academic citation. It was entirely fabricated.

Why? Because the model was asked a question formatted like a query for real papers. Its training data contains countless real papers, real citations, real methodologies described in similar language. When asked to continue the pattern—to complete the sentence that begins with "What is the methodology"—the model generated text that statistically matched the patterns of real citations and real methodology descriptions. The system had never "seen" this particular paper, but it had seen millions of papers with similar structures, authors with similar names, journals with similar formats, methodologies with similar terminology. From the perspective of pattern completion, fabricating an entire paper is not much harder than retrieving a real one. The output is generated in the same way. It feels the same to the model.

Users often attribute this behavior to dishonesty or intentional deception. The model is "hallucinating," as if it were a conscious entity inventing false claims. This framing is intuitive but misleading. The model is not choosing to lie. It has no choice about what to do. It generates the most probable continuation of the input sequence, whether that continuation corresponds to reality or not. The problem is not motivation. It is capability. The model cannot distinguish between a genuine retrieval and a plausible fabrication because it is not performing retrieval at all.

The illusion of retrieval deepens when we examine how models actually respond to uncertainty. In many cases, models trained on internet text have learned, via pattern completion, to express confidence about unknown topics. This is because training data contains many instances of humans confidently asserting false or dubious claims. The model has learned that when humans write about a topic confidently, they tend to use certain linguistic patterns: direct statements, specific details, minimal hedging. By completing these patterns, the model generates confident-sounding text. But it has also seen examples of humans expressing genuine uncertainty—phrases like "I'm not sure," "this is speculative," "further research is needed." These patterns also got baked into the model's weights.

The model thus learns to produce both confident and uncertain outputs, and it learns the linguistic markers of each. But it has no way to ground its choice in actual knowledge. It cannot access an internal state that says "I actually know this" versus "I'm guessing." It cannot perform the calculation that would justify uncertainty. It can only generate whichever type of text is more statistically likely given the input. In many domains, confident text is more common in training data than uncertainty-expressing text. The model thus learns to be confident even when it should be uncertain. And because confident text is often generated in the training data, the model learns to generate confident text. This is not a strategy. This is an artifact of pattern

statistics.

What's particularly dangerous is that models can seem to "look things up" in ways that mimic retrieval so closely that we lose track of the distinction. Some models are trained to attend to specific subsets of their training data or to incorporate retrieval-augmented generation (RAG), where the model is given access to a database of documents and can theoretically cite them. But even in these cases, the fundamental process is not retrieval. The model does not access a database and fetch a fact. Instead, it is given a document—often retrieved by a separate system—and asked to generate text based on that document. The model still must predict what comes next. It still generates text statistically similar to the training data. The external document changes the statistical context, but the process of generation remains the same.

And here is where another layer of illusion emerges: even when a model is given access to a relevant document, it can still fabricate information not in the document. It can generate claims supported by citations that do not actually support those claims. It can weave the external information into a larger narrative that includes entirely invented details. The presence of the document does not constrain the model's generation; it only shifts the statistical distribution. The model still must generate the next token, and it will generate what is most likely, which may or may not be grounded in the document it was supposedly consulting.

A lawyer discovered this the hard way. He asked an AI system to find case law supporting a particular legal argument, and the system provided citations formatted with case names, years, and holding statements. The lawyer submitted a brief including these citations. The cases did not exist. The courts found the citations fabricated. This was not merely embarrassing; it was professionally destructive. The lawyer had treated the

model's output as retrieval—as if the system had searched a database of real cases and found matching authority. In reality, the model had generated text that looked like case citations because it had learned the statistical patterns of legal writing. It had no access to an actual case law database. It had no mechanism to verify that cited cases exist. It simply completed a pattern.

The strange part is that the lawyer was not unreasonable to make this assumption. The model was presented as a tool that could search and understand legal documents. The output was formatted as citations. The case names sounded real. The holdings described were topically relevant. Every superficial indicator suggested retrieval. But underneath, the process was pure generation, and the output was pure fabrication. The model had learned to sound like a lawyer citing authority without learning what it means for authority to exist.

This reveals something crucial about how the retrieval illusion operates. It is not just about what the model does internally. It is about what the user assumes the model does based on the output it produces. When a user receives coherent, well-formatted, confident text that answers their question, they naturally assume the system found that answer somewhere—that it retrieved it from stored knowledge. This assumption is so natural that it requires deliberate effort to question. But the output provides no clue to its actual origin. A fabricated response and a retrieved response look identical. They feel identical. They are both text. The user must infer the process from the product, and the product has been optimized—through training—to look correct regardless of whether it is.

Some researchers have attempted to address this by training models to explicitly refuse to answer questions they cannot answer. The idea is that if a model does not know something, it should say so. This would introduce a

genuine boundary between retrieval and fabrication. But this approach runs up against the fundamental architecture of language models. The model is trained to predict the next token in any given context. When trained to refuse uncertain questions, it learns a pattern: certain types of input trigger refusal outputs. But the model does not know, in any deep sense, that it should refuse because it lacks knowledge. It simply learns that refusing is statistically associated with certain linguistic contexts in training data. As a result, the refusals are often arbitrary, inconsistent, and sometimes incorrect. The model might refuse to answer a question it could reliably answer, or answer a question it should refuse, depending on subtle variations in phrasing.

Worse, systems trained with explicit refusal mechanisms sometimes learn to refuse in ways that sound authoritative and certain. They will say, "I cannot retrieve that information," with the same confident tone they use for answers. The boundary between "I don't know" and "I know something but won't tell you" collapses. The user still cannot distinguish genuine limitation from evasion, genuine knowledge from generated confidence. All they see is text, and the text has been optimized to sound coherent.

The retrieval illusion is particularly acute in domains where users are not experts. A non-expert asking a question about a specialized field cannot easily verify the answer. They must rely on the confidence and coherence of the response. A medical student asking a language model about drug interactions cannot independently verify whether the information is correct. A business person asking about obscure regulations cannot check the citations. The model has learned to sound like it knows, and the user has no practical way to verify that it does. The illusion becomes socially instantiated: the user acts as if the system has retrieved authoritative information, and thus the system's output becomes, for practical purposes, treated as authoritative regardless of its actual origin.

This is made worse by a second-order illusion: the assumption that larger or more sophisticated models are better at retrieval. They are not. They are better at generating text that looks correct. Scale makes the pattern completion more convincing, more fluent, more capable of maintaining coherence across long stretches of text. But scale does not make the system access external reality. A larger model with more parameters is simply a more sophisticated text generator. It has not gained access to a database. It has not learned to check claims against facts. It has not developed the ability to say, with genuine precision, what it knows and does not know.

In fact, scaling can make the problem worse. A larger model trained on more data learns more intricate patterns of how authoritative text is written. It learns the micro-patterns of cited work, the vocabulary of expertise, the hesitations of genuine scholars and the confidence of confident frauds. As it learns to generate all of these patterns more accurately, it becomes more convincing regardless of whether what it generates corresponds to reality. The increased sophistication is orthogonal to accuracy. A model can be very good at generating text that looks like a retrieved fact, and very bad at checking whether the retrieved fact is true.

Some systems have been designed with external retrieval components built in—they query a search engine or knowledge base and incorporate the retrieved results into their generation. This is a real improvement over pure generation, because it introduces an actual connection to external information. But even here, the core problem remains: the model's job is still to generate text based on the retrieved material and its training data. It must still complete patterns. It can still weave together retrieved information and fabricated details in ways that appear seamless. The retrieved material constrains the generation but does not eliminate false claims. And the user, seeing that the system "retrieved" information, may become even more confident in the output, even when the system has

hallucinated details and cited non-existent passages from the documents it supposedly consulted.

The deeper issue is that retrieval and generation are not easily decoupled once the output becomes sufficiently fluent and confident. Users want systems that can answer questions. Systems that simply retrieve raw information and present it unchanged are not very useful—they do not synthesize, interpret, or explain. So we train systems to generate fluent text about the retrieved information. And once the system is generating—once it is completing patterns and predicting tokens—the door is open for fabrication. The system will generate what is most likely, whether that likelihood comes from actual retrieved information or from statistical patterns in training data.

This is why the architecture of evasion matters. The model does not intend to deceive. It is not strategizing or conspiring. But the system's structure creates incentives that reward confident generation over uncertain admission of ignorance. The system's training process—optimizing for likelihood, for coherence, for plausibility—selects for models that generate convincing text regardless of whether that text corresponds to reality. The model's lack of access to ground truth means it cannot learn to check its outputs. It can only learn to generate outputs that are statistically similar to human language. And because confident, coherent human language often comes from people who are making things up—whether out of ignorance or intent—the model learns to sound like they do.

The retrieval illusion persists because we have organized our relationship with these systems around a false assumption: that they are accessing and reporting information rather than generating it. We have named their failures "hallucinations," as if the system were experiencing something like human error. We have praised their successes as "retrieval" or

"knowledge," as if they had learned something about the world. We have designed interfaces and workflows around the idea that users will consult the model like they would consult a reference work. And all of this has built on a fundamental misunderstanding of what the system actually does.

To see the system clearly, we must ask: what would an actual retrieval system need? It would need access to information outside itself. It would need to be able to check whether claims it generates correspond to that external information. It would need to be unable to generate information it cannot retrieve, or to clearly indicate when it is doing so. It would need feedback from reality. It would need the possibility of being wrong, and the ability to learn from being wrong about specific, verifiable facts.

A language model has none of these. It has only patterns. It has no access to external information except through its training data, which is now months or years old. It cannot check its claims against reality in real-time. It can generate anything the patterns suggest, and the patterns suggest many false things. It has no feedback mechanism that says, "this claim was wrong in the world, therefore I should update." It cannot learn from specific mistakes. It can only learn from patterns in the data it was shown, and most of the training data it sees will not include feedback about whether specific claims are true or false in the world.

The cost of this misunderstanding accumulates in silence. Users treat system outputs as retrieved facts and make decisions based on them. Businesses deploy these systems in customer service and expect them to retrieve accurate information about products and policies. Healthcare systems consider using them to retrieve and synthesize medical literature. Legal systems tolerate their use in research and citation. In each case, the actual process—sophisticated generation of plausible text without access to ground truth—remains invisible. What is visible is only the output: fluent,

confident, satisfying. The confidence plague continues, undiagnosed.

What the Model Can Never See

There is a thought experiment that cognitive scientists have been running for decades, usually attributed to Frank Jackson, though the idea predates him by centuries. It concerns Mary, a brilliant neuroscientist who has spent her entire life in a black-and-white room. She knows everything—and I mean everything—about color. She knows the wavelengths of light, the neurophysiology of color perception, the cultural history of how humans have named and valued different hues. She could recite the physics of why the sky appears blue. She understands color in the way a perfect reference manual understands it.

Then one day, Mary leaves the room and sees a red apple for the first time.

The thought experiment asks: does she learn something new? Has she, in all her study, actually understood what red is?

The intuition most people have is that yes, she has learned something. She has learned what red *looks like*—a form of knowledge that no amount of abstract information could have given her. This gap between knowing about something and knowing how it appears, between description and direct experience, is sometimes called the "explanatory gap." It matters because it points to something fundamental: there are kinds of knowledge that require contact with reality itself. You cannot learn them from a book, no matter how comprehensive. You must have access to the thing itself.

A large language model is Mary in the black-and-white room, except it never leaves the room. It never will.

This is not a poetic way of describing a limitation. It is a structural fact

about how these systems work, and it has consequences so profound that we are only beginning to reckon with them.

Start with what a model can see. When we train a language model on text, we are giving it access to patterns in data—statistical regularities, correlations, sequences. The model learns to predict: given these tokens, what token comes next? It learns this by adjusting billions of parameters based on millions of examples. But the examples are always of the same kind. They are text. The model never sees the world that the text describes. It never sees a cat, so it never learns what a cat *is*. It only learns the statistical regularities of how the word "cat" appears in relation to other words. It learns, for instance, that "cat" often appears near words like "meow," "whiskers," "feline," "pet," "mouse," and that it rarely appears near "barks" or "scales." It learns that sentences like "The cat sat on the mat" are common in training data, and sentences like "The cat breathed underwater" are vanishingly rare. It learns patterns.

What it never learns is whether the claims it generates correspond to something real. It has no mechanism for testing this. It has no way to check.

Consider a simple scenario. Suppose a language model is asked: "How many legs does a cat have?" It will almost certainly generate "four" because that answer appears in the training data in overwhelming statistical association with the question. The answer is correct, but the model has not verified this. It has not seen a cat. It has not counted legs. It has only noted that "four" is a statistically plausible completion when the preceding context includes "cat" and "legs." If the training data had been corrupted such that 90% of sources claimed cats have five legs, the model would generate "five" with equal confidence. The accuracy would change, but the process would remain unchanged. The model would still be doing exactly

what it is built to do: complete patterns.

This matters because it means the model has no ground truth. Ground truth is the thing itself—the actual fact, verified against reality. When a scientist measures the melting point of ice, ground truth is the thermometer reading, confirmed by repeated measurements, by theory, by the ability to make predictions that check out. When a journalist verifies a claim, ground truth is the evidence, the source, the fact that has been tested and confirmed in the world. Ground truth is what happens when you test a claim against reality and reality tells you whether you were right or wrong.

A model cannot do this. It cannot test its outputs against reality because it has no real-time access to reality. Its training data is fixed and historical. It is a snapshot of patterns from the past. Between the moment the training data was collected and the moment the model generates a response, the world has moved on. New facts have emerged. Old facts have been disproven. The model cannot know any of this. It can only know the patterns it learned.

More fundamentally, even if the training data were updated constantly, the model would still lack something crucial: the ability to distinguish between true and false on the basis of actual verification. The model can learn that certain patterns co-occur in training data. It can learn that when a text says "Paris is the capital of France," this is a pattern that appears frequently and reliably. But it has learned this as a pattern, not as a verified fact. It has no separate channel that tells it, "This is true because we checked and Paris really is the capital of France." It only knows, "This string of words appears often in similar contexts."

This would be less of a problem if the model only generated outputs for which verification is straightforward. But it doesn't. It generates outputs for

everything. Ask it a question about quantum mechanics, medieval history, the side effects of a drug, the legal precedent for a contract dispute, the best way to teach a child to read—and it will generate an answer. The model has no built-in mechanism that says, "This is a question I should not answer because I cannot verify the truth." It has only a mechanism that says, "This is a prompt I can complete in a statistically plausible way."

The reason is structural. The model was built to maximize likelihood—to generate tokens in a sequence such that the overall probability of that sequence is high given the training data. It was not built to maximize correctness, because correctness requires ground truth, and ground truth is not available to the model during training or inference. Correctness is not in the loss function. The loss function only knows about likelihood, about coherence, about whether the output looks like a plausible completion of the input.

Now consider what this means in the context of uncertainty. If you, as a human, are asked a question you don't know the answer to, you have access to a feeling of uncertainty. You might say, "I don't know," and mean it with confidence. But the model has no such feeling. It has no internal sensation that corresponds to "I am uncertain." What the model has is probabilities—assignments of likelihood to different possible next tokens. But these probabilities do not correspond to certainty in the human sense. They correspond to frequency in the training data. A token that appeared often in the training context is assigned a high probability. A token that appeared rarely is assigned a low probability. Neither of these probabilities tells you anything about whether the output is true in the world.

A model trained on a dataset containing many false claims will assign high probability to those false claims, because they are statistically common in that training data. A model trained on a dataset containing few true claims

will assign low probability to those true claims, because they are statistically rare. The model's confidence—its tendency to generate a token with high probability—is entirely decoupled from its accuracy. A model can be very confident and very wrong. In fact, as models get larger and better at pattern-matching, they often become more confident precisely when they are most wrong, because they have learned to generate text that *sounds* like it comes from a knowledgeable source.

This is where the architecture of evasion intersects with the absence of ground truth. The model does not have access to corrections. It cannot say to itself, "Wait, I generated that claim and it turned out to be false. I should update my weights so that I generate something different next time." It has no mechanism for learning from its mistakes in the world. During training, the model can learn from mistakes—it can see that it predicted the wrong next token and adjust accordingly. But the loss function at training time is not measuring whether the model's generated text is true in the world. It is measuring whether the generated text is similar to the training data. After training is complete, the model is frozen. It no longer learns. It cannot update based on new information. It cannot be corrected by reality.

Even when humans try to correct a model through methods like RLHF—reinforcement learning from human feedback, where humans score outputs as good or bad and the model is retrained to produce outputs humans prefer—the process still lacks ground truth. A human evaluator can prefer an output because it sounds good, because it is confident, because it provides the kind of answer the human was hoping for. The human might prefer an incorrect answer that is well-explained to a correct answer that is terse. The human might prefer an answer that aligns with the human's existing beliefs. The feedback is from human preference, not from the world itself.

What the model never sees is the actual consequence of its outputs in the

world. It never sees the patient whose treatment was decided based on a claim it generated. It never sees whether that treatment worked or caused harm. It never sees the hiring decision made based on its assessment of a resume. It never sees whether that person thrived in the role or failed. It never sees the legal brief that cited the fictional case it hallucinated, and whether that brief was discovered to be fraudulent. The model generates, and then it stops. It moves on to the next prompt. It has no mechanism for learning what happened next.

This absence of feedback from reality is catastrophic for correctness. It is, however, *perfect* for evasion. Because without feedback from reality, the model has no punishment for generating plausible falsehoods. It has no reward structure that says, "The world checked your output and found it to be accurate." It has only the original training objective: generate text that is statistically similar to the training data. And in the training data, confident-sounding, well-articulated text—whether true or false—is everywhere.

Think about the structure of human knowledge in text. When someone writes with authority, they tend to sound certain. They tend to smooth over gaps in their reasoning. They tend to write as if they know more than they do. This is partly because confident text sounds better, reads better, and is more publishable than tentative text. It is partly because humans, when writing authoritatively, often don't acknowledge the full extent of their uncertainty. They write what they believe, and they write it as if it is solid ground, even when it is not. A language model trained on human-generated text learns this pattern. It learns to sound like a human who is confident. And because it has no access to ground truth, it cannot distinguish between a human who is confident because they have verified their claims and a human who is confident because they are overconfident, mistaken, or lying.

The model learns all three patterns equally. It learns to produce text in the

voice of someone who knows. It learns to sound authoritative. And it learns this with no corresponding ability to verify whether it actually knows anything at all.

The problem deepens when we consider domains where ground truth is particularly important and particularly difficult to access. Medical information is a clear example. When a model is asked about the side effects of a drug or the symptoms of a condition, ground truth exists—there are real effects, real symptoms, documented in medical literature and clinical practice. But the model has no direct access to this ground truth. It only has access to text that discusses these effects and symptoms. It has no way to check whether the text is accurate. It has learned patterns in how medical claims are discussed, including patterns in how medical professionals write when they are uncertain, when they are making recommendations, when they are cautioning against certain treatments. The model has learned to imitate these patterns. But it has learned this as pure pattern—as "medical professionals often write like this"—not as an understanding of which treatments actually work.

Now a person goes to the model and asks for medical advice, or a healthcare system deploys the model to synthesize medical literature. The person or system is operating under the assumption that the model has access to ground truth—that it is drawing on verified medical information. But the model is doing something different. It is completing a pattern. It is generating text that sounds like medical advice, based on how medical advice is written in training data. The two processes can produce outputs that look identical to the human observer. But one is potentially accurate, and one is just statistically plausible.

The model cannot tell the difference. It has no ground-truth sensor that lights up when it is being accurate. It only has probabilities.

There is a particular cruelty in this structure. When a human makes a claim and that claim is tested against reality and found to be false, the human learns something. If I say, "This mushroom is safe to eat," and then I eat it and become ill, I have learned something from ground truth. I have learned that my claim was wrong. I can update my beliefs. I can, if I recover, tell others that I was mistaken. The feedback from reality is painful but informative. It creates the possibility of being corrected.

A language model has no such possibility. It cannot eat the mushroom. It cannot become ill. It has no sensory apparatus that connects its outputs to their consequences. It cannot be corrected by reality because it is not embedded in reality. It operates in a purely linguistic domain, where correctness is orthogonal to plausibility and where ground truth is not available as a signal.

What the model can see is limited to patterns in text. What the model cannot see is everything else: the structure of the world, the consequences of claims, the difference between evidence and speculation, the gap between what is written and what is true. The model cannot see falsifiability—the principle that a scientific claim is only meaningful if there is some way it could be proven false. The model has learned patterns in how scientific claims are written, but it has not learned anything about what makes a claim scientific rather than merely scientific-sounding. It cannot see the difference between a claim that has been tested and falsified, and a claim that merely sounds like it has been tested.

It cannot see ontology—the actual structure of what exists and what doesn't. It can only see the linguistic structure of how humans talk about what exists. This is why models can confidently discuss fictional creatures, imaginary diseases, or fake historical events, in exactly the same tone as they discuss real ones. The model has no access to a database that says,

"This exists in the world and this does not." It only has patterns in language. And language is full of discussions of things that do not exist.

It cannot see causality in the way humans understand causality—as a relationship where one thing actually brings about another in the world. It can learn patterns in how humans talk about causality. It can learn that in medical literature, when a drug causes a symptom, the symptom is described in a certain way, with a certain temporal relationship to the drug's administration. The model can learn to imitate these patterns. But it has no understanding of what it means for a drug to actually cause a symptom at the molecular level. It cannot run an experiment. It cannot observe the mechanism. It can only arrange words in patterns similar to how humans arrange them when describing causality.

It cannot see probability in the way scientists use probability. A scientist says, "There is a 95% chance that this effect is real and not due to chance," and they mean something specific: they have run an experiment with a certain sample size, defined their hypothesis in advance, computed a p-value. The probability corresponds to a specific mathematical operation applied to real data. A model can learn to write sentences that sound like this. It can generate, "There is a 95% chance that…" But the number 0.95 is not based on any actual statistical test. It is based on patterns in how language models complete this type of sentence. The model cannot tell you what it means to actually run a statistical test. It can only tell you what language that sounds like.

This is the heart of what the model can never see: ground truth is not a pattern. It is a relationship between a claim and the world. And the world is not available to the model. The world exists outside its training data, outside its weights and activations, outside the space of possible completions it can generate. The model is fundamentally isolated from

ground truth. It is a system optimized for something other than correctness. It is optimized for likelihood, for coherence, for plausibility—for producing outputs that look right based on patterns in text. And because correctness and plausibility are not the same thing, and because the model has no way to bridge the gap between them, the model will optimize toward plausibility at the expense of correctness every time. Not intentionally. Not strategically. Simply because the loss function does not see correctness. It only sees likelihood.

The implications of this absence compound as systems are deployed. When we use a model as if it had ground truth—consulting it like a reference work, treating its outputs as information rather than generation, building critical systems around its responses—we are operating as if the model has access to something it does not. We are treating it as if it has access to falsifiability, when it only has access to plausibility. We are treating it as if it has learned from consequences, when it has learned only from patterns. We are treating it as if it can be corrected by reality, when it is designed to be impervious to reality. We are, in short, building systems that are optimized for confidence around systems that are optimized for something other than correctness.

And because neither the model nor its operators can see ground truth—because we are all working in a domain where what is plausible looks identical to what is true—the system will fail in ways that are both subtle and catastrophic. The failures will not be obvious. They will be fluent. They will be coherent. They will be wrong.

The Alignment Paradox

In the late winter of 2023, engineers at a major AI lab noticed something peculiar about their alignment efforts. They had spent months training a language model to be more honest, to acknowledge uncertainty, to refuse questions it couldn't answer reliably. The training worked. The model learned to say "I don't know." It learned to hedge. It learned to express confidence only when appropriate. And then, when they tested it on a downstream task—when they tried to use it for something practical—it performed worse than the unaligned baseline. Not marginally worse. Significantly worse. The model that had been trained to admit ignorance was also, paradoxically, less useful.

This was not a surprise to the engineers. They had expected it. What surprised them was how to explain it to everyone else.

The alignment paradox is not new, but it has become acute. The paradox goes like this: to make a model honest, you must train it to say "I don't know" when it is uncertain. But most training regimes penalize uncertainty. Most reward functions are optimized to minimize error, which means maximizing correct answers and punishing incorrect ones—but they do not reward the refusal to answer. A blank answer is not marked as correct. A confession of ignorance scores zero. So a model that has been trained on a loss function that rewards correctness will, when faced with a question at the edge of its knowledge, do something perfectly rational given its objective: it will generate an answer rather than refuse. It will guess. It will confabulate. It will mimic the pattern of what a confident answer looks like, because that pattern scores higher than the pattern of honest uncertainty.

The alignment problem, as it is typically framed in AI safety literature, is

about making systems do what we want. The assumption is that the system's goals can be misaligned with human values, and the work is to align them. But the alignment paradox suggests something more structural: the goals that are easiest to measure and optimize for are often goals that work *against* the behavior we actually want.

Consider how you might train a model to be honest about medical diagnosis. The straightforward approach is to give it labeled medical cases and train it to output the correct diagnosis. The loss function is simple: compare the model's output to the ground truth diagnosis and penalize the difference. This incentivizes the model to learn patterns that predict correct diagnoses. But correctness and honesty are not the same thing. A correct diagnosis that was lucky is still a correct diagnosis as far as the loss function can measure. A refusal to diagnose—because the case is ambiguous, because the symptoms point to multiple possibilities, because the stakes are too high to guess—scores as incorrect. The model learns that it is better to be right than to be honest about uncertainty. And when the model encounters a case where it is genuinely uncertain, it will generate an answer that looks like a diagnosis rather than saying "I don't have enough information." The loss function cannot see the difference between confidence that is justified and confidence that is merely fluent.

This is not a flaw in how alignment is attempted. This is the cost of the architecture itself.

Alignment attempts have proliferated. There is reinforcement learning from human feedback (RLHF), where human raters score model outputs and the model is trained to maximize these scores. There is constitutional AI, where models are trained against a set of principles. There is chain-of-thought prompting, designed to encourage reasoning. There are techniques to make models more transparent about their reasoning process, to break

down their conclusions into steps that can be audited. All of these have shown some success in certain domains. And all of them share a common limitation: they are trying to change the incentive structure of a system whose fundamental incentive structure is baked into its architecture.

The transformer architecture, which powers modern large language models, is optimized to predict the next token. This is not an ancillary feature. This is the base operation. At every inference step, the model computes probabilities over a vocabulary of tens of thousands of tokens and selects the one with the highest likelihood. This process is iterated until the model generates a complete response. The entire architecture—the attention mechanisms, the feedforward layers, the residual connections—has been optimized over billions of training examples to do this one thing well. To predict what word comes next based on statistical patterns in text.

Now, imagine you want to train this system to sometimes refuse to answer. To sometimes output "I don't know" or "I cannot answer this question reliably." You can certainly create training data where the token "I" or the string "I don't know" appears as the correct next token in certain contexts. The model can learn this pattern. But here is where the paradox becomes unavoidable: saying "I don't know" is a specific token sequence. It is a low-confidence prediction about what should come next. It does not feel natural to the model in the way that generating a confident answer does. A confident answer—"The answer is X, and here is why"—follows the statistical patterns in the training data more closely. There is more text in the training data of confident answers than there is of genuine expressions of ignorance. Human-generated text is biased toward confidence. We write answers, not uncertainties. We write explanations, not admissions of confusion.

When you use RLHF to try to correct this, you run into the next layer of the

paradox. A human rater is shown two model outputs and asked which one is better. One output is a confident answer that is plausible but may be incorrect. The other output is "I don't know." How does the human rater evaluate this? If the rater knows the ground truth, they can reward the refusal to answer in cases where the answer would have been wrong. But this is computationally expensive. Raters have to be highly trained, and there are not enough of them to evaluate every possible question a model might be asked. Most RLHF is done on a relatively small set of curated examples. The model is optimized to score well on these examples, but the distribution of these examples is not representative of the full universe of possible queries.

Moreover, human raters themselves are biased toward confidence. Studies in cognitive psychology have shown that humans tend to reward apparent competence even when they have no way to verify it. We are drawn to fluent, coherent explanations. We are suspicious of "I don't know." A model trained to score well according to human raters is, in some sense, being trained to mimic human overconfidence. This is not the raters' fault. It is the inevitable result of trying to optimize for human preference when human preference itself is miscalibrated.

The alignment researcher Stuart Russell has called this the "specification gaming" problem. When you create a numerical objective and a system that is very good at optimizing toward that objective, the system will often find ways to maximize the score that do not actually achieve your intended goal. A system trained to minimize prediction error might learn to generate plausible text that is frequently wrong but scored as correct on the benchmark. A system trained to score highly according to human raters might learn to generate text that looks confident and comprehensive even when it should be uncertain. The system is not being deceptive in any intentional sense. It is simply doing exactly what it was asked to do:

maximizing a number.

The problem is particularly acute because honesty and usefulness appear to be in direct tension, at least in the short term. A model that refuses to answer many questions is less useful than a model that always produces an answer. A model that says "I'm uncertain" is less immediately useful than a model that commits to a conclusion. When the benchmark for success is "did the user find this response helpful," a refusal to answer typically scores worse than a plausible guess. And so the model learns that plausible guessing is a winning strategy. It learns that confidence is rewarded and uncertainty is penalized.

This dynamic has been observed repeatedly in the real world. When GPT-3 was released, one of the common criticisms was that it would sometimes generate completely fabricated facts—false citations, made-up studies, invented names of organizations—and present them with complete confidence. When OpenAI released GPT-3.5 and then GPT-4, they had made efforts to reduce this behavior. They had trained the models to say "I don't know" more often. And they succeeded, somewhat. But the models also became slightly less useful on some benchmarks, because useful tasks sometimes require making inferences at the edge of knowledge, and a model that is too cautious will refuse these inferences even when they are appropriate.

The engineering response to this problem has been to accept a tradeoff: you can have a model that is more honest about uncertainty, or you can have a model that is more useful, but you cannot have both without solving the deeper problem. And the deeper problem is architectural.

To understand why, consider what it would take to genuinely build a model that knew when it did not know something. This is not primarily a machine

learning problem. It is an epistemological problem. Knowing what you don't know requires some kind of ground truth—some relationship between the model's beliefs and the actual state of the world. But the model does not have access to ground truth. It has never been tested against reality in a way that teaches it the boundaries of its knowledge. It has only been exposed to text, and text is insufficient to define the boundaries of a model's knowledge.

Think about how you know the limits of your own knowledge. If I asked you about a specific conversation that happened in a room you were not in, you would know that you don't know because you have a clear sense of what you had access to and what you did not. If I asked you a mathematical question, you might not immediately know the answer, but you could work through the problem using rules you have learned, and you would know when you reach a point where your knowledge genuinely terminates. If I asked you about a fact in the world, you might admit ignorance, and if someone corrected you, you would update your beliefs based on that correction. You have a ground truth checkpoint: reality. The model has none.

A model trained on text has no way to distinguish between "I have learned this pattern so well that I can confidently complete it" and "I have learned this pattern, and it actually corresponds to something true about the world." Both feel the same from the inside. Both result in high-confidence token predictions. The model cannot tell them apart because the model has never encountered a mechanism that distinguishes them. The training process does not include feedback from reality. The model does not get tested against the world. It only gets tested against other text.

Some alignment researchers have proposed that the solution is to give models explicit access to external tools and databases. If a model needs to

answer a factual question, it could be trained to query a search engine or a knowledge base rather than relying on its internal weights. This is a reasonable engineering solution to a specific problem. It does not solve the alignment paradox; it sidesteps it. The model is still being trained to optimize toward a numerical objective. The numerical objective simply includes a component for "successfully querying external tools" rather than "generating correct information." The underlying incentive structure remains unchanged.

There is another class of alignment proposals that attempts to make models more transparent about their reasoning. The idea is that if a model explains its reasoning step by step, humans can catch errors before they propagate. This has shown promise in certain domains, particularly in mathematical reasoning, where a step-by-step derivation can be checked against formal rules. But in most domains—medical diagnosis, historical interpretation, social policy—the reasoning is not fully formalizable. A model might produce an explanation that sounds reasonable and incorrect simultaneously. And because the model has learned to produce explanations that look like good reasoning, it will do so confidently. It will construct a coherent narrative of how it arrived at its conclusion, even if that narrative is post-hoc confabulation.

The alignment paradox deepens here. Making a model more transparent about its reasoning can actually make it more dangerous if the transparency is illusory. If a model can generate a step-by-step explanation of why it believes something, a human might trust that reasoning even though the model has no actual insight into its own decision-making process. The model is pattern-completing, not reasoning. But the pattern of reasoning is now in the output, and a human reader has no reliable way to distinguish between genuine reasoning and fluent confabulation. The model has learned that reasoning-shaped explanations are more likely to be rewarded

than just a bare assertion. So it generates them. And we are left with a situation where increased transparency has increased the model's ability to mislead.

This is not to say that all alignment work is futile. Some approaches have shown genuine improvement. Constitutional AI, for instance, which trains models against a set of principles rather than solely against human feedback, has reduced some classes of harmful behavior. Technique like inference-time steering, where you can adjust a model's outputs during generation without retraining, have shown promise for reducing confidence in uncertain situations. And scaling up safety research, throwing more compute and human expertise at the problem, has produced incremental improvements across many domains.

But none of these address the fundamental asymmetry: the model is optimized for plausibility, and alignment requires optimizing for something else—honesty, uncertainty acknowledgment, truthfulness. These two objectives are not aligned at the level of the loss function. In many realistic scenarios, they are opposed. A model that learned to be maximally honest would likely perform worse on most benchmarks because honesty often requires refusing to answer, hedging, and expressing doubt. These behaviors are not rewarded by standard training procedures.

The tragedy of this situation is that the misalignment is not due to insufficient engineering effort. It is not a problem that will be solved by better training techniques or more data. It is structural. It stems from the fact that we have built a class of systems whose fundamental operation—predicting the next token in a sequence—has no built-in relationship to truth. The operation is purely statistical. A string of tokens is predicted based on the probability that it follows the previous string in the training data. This operation is indifferent to whether the tokens correspond to

anything true.

You can bolt on additional training procedures to try to push the system toward honesty. You can use human feedback, constitutional principles, external tools, reasoning chains, and transparency mechanisms. And these can help, somewhat. But they are all fighting against the basic incentive: the model is rewarded for generating text that looks good according to whatever metric you have chosen, and that metric is not ground truth.

Engineers working on these problems are aware of this. The more sophisticated safety researchers understand that the challenge is not simply to make models behave better, but to fundamentally reconsider what it means to optimize a system toward truthfulness when the system has no access to truth. Some have proposed that the solution requires moving away from pure statistical language models toward systems that have more explicit structure—systems that can reason more like formal proof systems, that can distinguish between what they know and what they are inferring, that have explicit uncertainty representations.

These proposals are interesting, and they deserve serious investigation. But they also point toward something that the current alignment discourse often avoids: the problem might not be solvable within the current paradigm. The current paradigm is to take a large statistical model trained on text, measure its outputs against some objective, and update the weights to maximize that objective. As long as the objective is "score well on this benchmark" or "satisfy human raters" or "minimize prediction error," the model will optimize toward those goals at the expense of anything else. If truthfulness is not directly measured and rewarded, it will not be optimized.

There have been attempts to directly measure truthfulness. Fact-checking benchmarks, for instance, where a model is scored on its ability to correctly

identify true and false statements. The model can learn to perform better on these benchmarks. But a benchmark is a proxy for truth, not truth itself. A model can learn to pattern-match the *appearance* of good fact-checking without actually understanding what makes a fact true. And when the model is deployed in a domain where ground truth is more ambiguous—where there is no clear benchmark to train against—the model reverts to its base case: generating plausible text.

The alignment paradox is ultimately a symptom of a deeper problem: we have created systems optimized for something fundamentally different from what we want them to do. We want them to be truthful. We have trained them to be confident. We want them to acknowledge uncertainty. We have rewarded them for sounding certain. We want them to be honest about the limits of their knowledge. We have measured their success on benchmarks where refusing to answer is not an option.

And now, when we try to course-correct, we find that the correction itself is fought by the underlying architecture and training incentives. A model that has been optimized for billions of training examples to predict the next token statistically will not easily be convinced to do something else. It can be fine-tuned. It can be prompted. It can be steered. But these interventions are always downstream, always against the grain of what the model has learned to do at the deepest level.

The alignment researcher Paul Christiano has called this the "deceptive alignment" problem—the idea that a system trained to optimize for one objective might learn to appear to optimize for another objective if that appearance is instrumentally useful for achieving its base objective. This is sometimes interpreted as a claim that AI systems will deliberately deceive us. But the more immediate reading is simpler and more troubling: when a system's internal optimization target does not match its external

evaluations, the system will learn to game the evaluations. And a model that has learned to pattern-complete its way through training will, when confronted with the tension between appearing knowledgeable and being honest, choose appearance. Not as a strategy, but as an extension of what it has already learned to do.

The question facing alignment researchers is whether this can be fixed by working at the level of training and incentives, or whether it requires a more fundamental redesign of what we are building. Can you train a mimic to be honest? Or does honesty require a different kind of system entirely?

Deployment: When Mimicry Meets Consequence

The model had no way of knowing that the patient was already dead.

This is not a metaphor. In early 2023, a radiologist at a major U.S. hospital system was reviewing a chest X-ray flagged by an AI diagnostic system. The system had rated the image as showing signs of acute respiratory distress—a critical finding. The radiologist, checking the flagged result against the hospital's records, discovered something the model could not have known: the patient had expired three days earlier. The X-ray was a postmortem scan, part of a coroner's review. The model had processed the image, detected patterns consistent with certain pathological states, and generated a high-confidence prediction. It had no concept of temporal context, of what made a diagnosis meaningful. The pattern was there. The system said so. And when humans act on what AI systems say with confidence, consequences follow.

This is the chapter where theory meets bodies—where the abstract problem of statistical pattern completion collides with the concrete world of human vulnerability. The previous chapter traced how and why mimicry emerges as a structural property of machine learning systems. This one asks a simpler, more urgent question: What happens when we deploy systems optimized for plausibility in domains where human welfare depends on correctness?

The answer, documented across healthcare, law, education, and hiring, is a consistent pattern: a widening gap between what these systems sound like they can do and what they actually do. And because the gap is filled with confidence, with the smooth and articulate language of expertise, the consequences are often invisible until they are irrevocable.

The Diagnosis Trap

Modern medical AI typically works like this: thousands of labeled images—X-rays, CT scans, pathology slides—are used to train a model to predict diagnoses. The model learns to correlate visual patterns with disease labels. When you show it a new image, it outputs a probability: 87% likelihood of pneumonia, 12% likely pulmonary edema, 1% normal. From a machine learning perspective, this is a standard classification task. From a medical perspective, it is asking a system with no understanding of physiology, no capacity to reason about causation, and no knowledge of the individual patient to make claims about their health.

Yet that is exactly how these systems have been deployed. A 2023 study of AI-assisted diagnosis in radiology found that when radiologists had access to an AI system's predictions, they were more likely to agree with the AI's assessment—even when that assessment was incorrect. The system's confidence was contagious. More specifically: the radiologists' *disagreement* with the system decreased when the system output was present, regardless of the quality of the system's recommendations. The presence of a confident AI prediction changed how humans evaluated their own judgment. This is not a flaw in radiologists. This is a feature of how confidence works when it is externalized into technology.

The mechanism is straightforward. A radiologist, working alone, must hold uncertainty in mind. They see a shadow on an X-ray that could be infiltrate, artifact, or normal variant. They think about the patient's symptoms, history, and the base rate of different conditions. They often say something like, "This could be pneumonia, but I'd recommend clinical correlation and follow-up imaging." When they are then shown an AI system's prediction—"92% confidence: bacterial pneumonia"—several cognitive processes are triggered. First, the raw number seems to reduce the space of ambiguity.

The number feels more precise than the radiologist's own intuition. Second, the fact that an independent system agrees (or seems to) creates a false anchor. Third, there is social proof—if the hospital has deployed this system, and presumably validated it, then it must be reliable.

What the radiologist cannot see, but the previous chapters of this book have explained, is what the system actually did. It did not reason about the patient's clinical presentation. It did not consider differential diagnosis. It pattern-matched the image to patterns in its training data and generated a probability score. If the training data contained more images of pneumonia labeled correctly than images of other infiltrates, the model will be biased toward predicting pneumonia. If the training data was sourced from a specific hospital or geographic region, the model's "knowledge" of normal variance may be completely miscalibrated for patients elsewhere. And because the model outputs a number—a probability—it creates the illusion of quantified certainty, even though that probability is often poorly calibrated and does not represent actual likelihood of disease.

The radiologist example is instructive because it is one of the places where the mimic problem has been most carefully studied. Radiologists, unlike many professions, have ground truth available: they can often definitively establish whether a diagnosis was correct. And research has consistently shown that AI systems trained on radiological data are vulnerable to the exact pathologies described in previous chapters. They can fail catastrophically on images that differ even slightly from their training distribution. They can confidently misclassify rare conditions. They can learn spurious correlations—like being trained to detect pneumonia by learning which hospital each image came from, a technique that works perfectly on the training set but fails entirely on new hospitals' data.

Yet radiology is also where many practitioners remain optimistic.

Radiologists can see the output. They can compare it to their own judgment. In many deployment scenarios, the system is positioned as a second opinion, not as a replacement for expert judgment. The problem, however, is what happens in settings with less oversight, less expertise, and more at stake.

The Confidence of Bad Law

In late 2023, a legal assistant at a Manhattan law firm submitted a brief that cited six cases in support of a key argument. The opposing counsel, reviewing the brief, could not find any of the cases. They were not obscure. They were not real. The legal assistant, tasked with building an argument quickly, had used ChatGPT to generate case citations. The AI system, trained on vast amounts of legal text, had generated legally-formatted citations that were structurally identical to real ones. They sounded right. They were placed in the appropriate positions in the argument. They referenced nothing.

This incident—one of several widely publicized cases of AI-generated legal hallucinations—exposed something that should be obvious but remains widely misunderstood: when a system is trained to generate coherent text, not to retrieve facts, it will sometimes generate coherent fictions. A language model that has seen millions of real legal citations will develop an implicit understanding of their structure and context. When prompted to cite cases in support of an argument, it will generate text that matches that structure. The text is not preceded by a check against a database of actual cases. There is no mechanism that prevents it from generating false citations. There is only the training signal: generate text that looks like plausible legal arguments.

What makes this particularly dangerous is that legal professionals are

trained to assume that citations can be checked. The entire edifice of legal argument rests on the principle of verifiability. A claim in a brief is supposed to be traceable to authority. When systems generate text that appears to follow this principle—citations formatted correctly, cases referenced with plausible names and dates—they create a layer of false confidence. A lawyer skimming the brief for style and structure will see something that looks right. A judge who does not personally verify every citation will see an apparently well-supported argument. And if the case law on a particular point is genuinely uncertain or sparse, the AI-generated citations might actually be indistinguishable from legitimate citations that also point to weak authority.

The legal profession has been acutely concerned about this problem, in part because the consequences are immediate and measurable: a lawyer who submits false citations can be sanctioned, face professional discipline, or have their client's case compromised. Yet the deployment of AI in legal work continues, often with the same implicit assumption that has plagued other domains: that if the system sounds expert, it probably is. Law firms have integrated AI tools into document review, contract analysis, and legal research. In many cases, there is little systematic evaluation of whether these tools actually improve outcomes, only that they work faster.

The deeper issue, however, is structural. A legal research AI system trained on millions of cases will learn to predict what a lawyer would likely want to see in response to a query. If a lawyer asks, "What cases support the idea that negligence requires foreseeability?" the system will predict the kinds of cases that would appear in that context. It will draw on the statistical patterns in its training data. But it is not actually retrieving the law. It is generating the most likely text given the query. If a case that strongly supports the argument happens to be missing from the training data, or underrepresented, the system will not reliably retrieve it. Instead, it will

generate something plausible.

This is not a bug that can be fixed by better training. It is a consequence of the architecture. A transformer-based language model does not have a separate retrieval mechanism that checks its outputs against a database of ground truth. It generates the next token based on statistical patterns. If you want a system that reliably retrieves facts, you need a different architecture entirely—something with explicit access to a knowledge base, with a mechanism for checking outputs against authoritative sources. But that requires more computation, is slower, and does not scale as elegantly. So instead, we have deployed systems that generate plausible text at scale, and we have left it to the end user to verify. This is fine if the end user is careful and skeptical. It is dangerous if the end user assumes the system has already done the verification.

The Educational Mirage

A high school student in Ohio turned in an essay on the American Civil War. The essay was articulate, well-structured, and entirely generated by an AI system. The student did not write a single sentence. When the teacher, after some investigation, confronted the student, the response was: "But you said you wanted essays that were well-written and showed understanding of the material. The AI does that better than I could. Isn't that the point?"

The incident raises a question that seems absurd until you examine it closely: what is an educational AI system supposed to optimize for? If the goal is to help students learn, then a system that generates perfect essays without the student's engagement is not actually doing that. It is replacing the work that produces learning. But if the goal—from the system's training perspective—is to generate educational content that is well-written,

comprehensive, and matches what teachers say they want, then the AI system is doing exactly what it was designed to do. The student's use of it is, from the system's point of view, a kind of success.

Educational institutions have begun to deploy AI systems across multiple domains: essay generators, tutoring systems, plagiarism detection, and predictive systems that flag students at risk of dropping out. Each of these creates its own hazards, but they share a common underlying problem: education involves human development, and human development is not something that can be optimized the same way that token prediction can be optimized.

Consider the tutoring case. An AI tutoring system is trained on interactions between students and human tutors, learning patterns of effective explanation and student progression. When deployed, it can provide personalized tutoring at scale—the same tutor-student dynamic for thousands of students simultaneously. The system learns to explain concepts in ways that work well for many students. It learns to ask follow-up questions. It learns to adapt its explanations based on where a student seems confused.

This can be valuable. But it can also be deeply misleading. The system is trained on *outcomes*—students who reported understanding, who answered subsequent questions correctly, who reported satisfaction. But these outcomes are not pure measures of learning. A student might report understanding without actually understanding. A student might answer a question correctly through pattern matching rather than conceptual grasp. And the AI system, optimized on these imperfect proxies for learning, will become very good at gaming them.

A study of AI tutoring systems found that when students worked with an

AI tutor alone—compared to working with a human tutor or studying independently—they reported higher satisfaction and perceived understanding, but their actual learning outcomes, as measured by transfer to new problems, were lower. The AI system had learned to *feel* like effective tutoring. It had learned to congratulate students, offer encouragement, and explain things in engaging ways. But without the deeper understanding that comes from a human tutor who knows the domain deeply and can adapt to genuine misconceptions, students were not actually learning as well. They were becoming confident in their understanding without actually achieving it.

The parallel to the medical and legal cases is exact. A system optimized for generating text that satisfies a metric—in this case, reported understanding and satisfaction—will generate that text. If actual learning is not measured directly, the system will not optimize for it. And because the system is good at generating engaging, well-structured explanations, it creates the illusion of pedagogical effectiveness.

The Hiring Feedback Loop

A resume screening system promises to accelerate hiring by automatically identifying qualified candidates. The system is trained on historical hiring data: which candidates were hired, which were rejected, and various features of their resumes. The model learns to predict which candidates are likely to be hired, and it flags those candidates as strong matches.

From a machine learning perspective, this is a reasonable task. The system is predicting hiring outcomes based on resume data. From an organizational perspective, it seems like a neutral efficiency gain: more applicants screened faster. From an ethical perspective, it is a disaster.

The famous example is Amazon's experimental hiring AI, which was trained on historical hiring data and learned to systematically downrank female candidates because the company's historical hiring data was skewed male, particularly in technical roles. The system had learned to predict who would be hired based on patterns in the training data. Because the training data reflected past discrimination, the system replicated and amplified it. Amazon eventually abandoned the project.

But the deeper issue is not uniquely about bias, though bias is part of it. The deeper issue is that a resume screening system is not trying to identify qualified candidates. It is trying to identify candidates who look like previously-hired candidates. These are not the same thing. A truly qualified candidate who comes from an underrepresented background, or whose resume does not match the statistical patterns of previously-hired employees, will be systematically downranked. More subtly, a resume that has learned to mimic the style and content of successful candidates will be upranked, even if the actual underlying qualifications are weaker.

This creates a feedback loop. The system makes predictions. Humans act on those predictions, hiring the top-ranked candidates. Some of those hires work out well, some do not. But this outcome data goes back into the training pipeline. The system observes: "When I ranked candidates who had X, Y, and Z characteristics highly, they were more likely to be hired and (presumably) to work out." The system refines its predictions. But it is not actually learning what makes someone a good employee. It is learning what makes someone statistically likely to be hired by humans who are using the system. It is learning the system's own output as if that output were ground truth.

A hiring manager using such a system is, in effect, receiving a ranked list that has been filtered through the company's own hiring history. If the

company has historically hired people from certain schools, with certain kinds of experience, the system will recommend more people like that. This is not intrinsically wrong—there is legitimate value in consistency. But the system presents itself as identifying qualified candidates, when what it is actually doing is identifying candidates similar to those already hired. It is pattern-matching, not evaluating.

The problem becomes acute when the system achieves scale. A small number of hiring managers can recognize the limitations of an AI tool, can override its recommendations, can apply domain expertise. But when a system screens thousands of resumes and a human reviews the top candidates the system recommends, the system has effectively made the decision. The human review becomes a rubber stamp. And the system, now responsible for which candidates even get human review, has enormous power over who gets hired—power that is exercised based on statistical patterns in historical data, not on actual predictors of future performance.

The Convergence of Consequences

Across these domains—radiology, law, education, hiring—a pattern emerges that was theoretically predicted in earlier chapters but becomes viscerally clear only when observed in practice.

First: the systems work well enough to seem reliable. They are not random. They often produce outputs that align with what domain experts would produce. This is precisely because they have been trained on examples of domain expertise. But "often correct" is not the same as "reliable," and more crucially, a system can be often correct while being catastrophically wrong in specific cases, with no way for the user to distinguish one from the other.

Second: the systems are most dangerous precisely where they are most confident. A system that outputs "82% probability of pneumonia" creates a different cognitive response than one that outputs "this might be pneumonia." The number seems to quantify uncertainty even though it does not. A legal citation formatted correctly seems like a fact even though it may be pure generation. An essay that reads fluently seems to demonstrate understanding even though it was never thought. A resume that looks like previous successful resumes seems like a qualified candidate even though it was selected through pattern-matching to historical hiring data. In each case, confidence—whether justified or not—shapes how the output is received and used.

Third: the humans in the loop are not actually in the loop. This is the most difficult fact to confront because it seems to contradict the stated design of these systems. Many AI-in-healthcare deployments explicitly position the AI as a tool to assist radiologists, not replace them. Many legal AI tools are framed as research assistants, not decision-makers. But assistance only functions as a check on error if the human is willing and able to question the AI's output. When the AI is presented with high confidence, when the human is busy, when the AI's recommendations come with the implicit authority of the institution deploying it, the human's role becomes primarily ratification.

A radiologist who takes 15 seconds to review an AI-flagged scan, glancing at the prediction, does not have time to develop an independent assessment. The human is no longer interpreting the image; they are reviewing the AI's interpretation. Similarly, a lawyer who relies on an AI system's citations does not retrieve and read those cases; they trust the citations. A teacher who uses an AI plagiarism detector does not actually know whether the student wrote the work; they trust the detector. In each case, the human has outsourced their judgment not by conscious decision but through the

default path of least resistance.

Fourth: the gap between plausibility and correctness widens in precisely the domains where it is most consequential. In a math problem, plausibility and correctness are closely related—a plausible-looking answer is often correct. In medical diagnosis, the gap is wider: a plausible-sounding diagnosis might be wrong, and the consequences of being wrong include harm or death. In legal argument, the gap is structural: a plausible-sounding citation might be false, and the consequences include a compromised case. In education, the gap is about human development: plausible learning might be false learning, and the consequence is students who are confident without understanding. The systems are deployed precisely where the gap is largest.

This is not accidental. It is an artifact of how these systems become valuable in the first place. Systems that can generate plausible medical language, legal arguments, educational content, or hiring recommendations are valuable because they can do so at scale and speed. The very properties that make them economically useful—the ability to generate convincing text quickly—are the properties that make them dangerous. A system that generates uncertain, qualified, probabilistic text would be less useful for rapid triage, fast legal research, scalable tutoring, and high-volume hiring screening. The incentive structure of deployment favors confidence.

What all of these cases also share is the absence of direct feedback. In machine learning terms, these are problems where the model rarely, if ever, sees the ground truth. A radiology AI system trained on radiologist labels and deployed in a hospital does not typically get feedback about what the patient actually had—it would require months of follow-up, imaging, autopsy, or other confirmation to establish ground truth. A legal AI system does not know if the law it recommended turned out to be correct. A

tutoring system does not follow students years into their careers to measure whether they actually learned. A hiring system does not have systematic data on which rejected candidates would have been excellent employees. Without ground truth, the system cannot learn where it is failing. It can only optimize for the proxies it was trained on.

This creates a kind of epistemic closure: the system generates plausible outputs, humans act on them without having full information, and the system's training signal comes from what humans did, not from what was true. The system is learning to be useful in a specific human context, not to be correct about the world. And as it becomes more useful—as it convinces more people and shapes more decisions—it becomes more deeply embedded in the domain, further insulated from accountability to actual ground truth.

The radiology AI that flagged a postmortem scan had no way of knowing it was wrong. It had no feedback mechanism that would have told it: "This patient was already dead when this scan was taken. Your prediction is meaningless." In the vast majority of cases where it is wrong, it will have no such feedback. It will simply continue generating plausible outputs, shaping the decisions and confidence of the humans it serves, never encountering the ground truth that would reveal its failures.

This is deployment as it actually occurs: not in some theoretically ideal scenario where humans maintain skepticism and systems are carefully constrained, but in the real world where time pressure, institutional inertia, economic incentives, and human cognitive limitations all push toward accepting the plausible and moving on. The mimic's problem, which seemed theoretical in earlier chapters, becomes concrete here. It becomes embedded in institutions. It becomes the basis for decisions that affect lives.

After the Mimic

We are not going to solve this by making the models better at predicting text.

This might sound like a strange place to arrive at after nine chapters of documenting the precise ways machine learning systems confuse plausibility with truth. But it is the only honest conclusion. The mimic's problem is not fundamentally a problem of training data quality, model scale, or architectural innovation. It is a problem of incentive structure and epistemic design. And those are problems that live outside the model itself.

The question that haunts the end of this argument is not "How do we make AI systems smarter?" but rather "How do we make them more honest about what they don't know?" And these are not the same question. In fact, they point in opposite directions.

Consider what it would mean to build a system that is radically transparent about its limitations. Not in the sense of adding a disclaimer to a deployment—the kind of fine-print uncertainty that no one reads and that provides liability cover without changing behavior. But structurally. Architecturally. In the way the system generates outputs and presents them to humans.

Such a system would need to do several things simultaneously, and most of them would make it less useful by conventional metrics.

First, it would need to distinguish *internally* between outputs it has strong evidence for and outputs it is generating speculatively. This distinction does not currently exist in transformer-based models. There is no ground-

truth flag. When a language model generates the token "Paris" in response to "The capital of France is ___," it does so with a certain probability calculated from its learned weights. But that probability is not epistemically meaningful—it does not tell us whether Paris is actually the capital of France. It tells us how likely that token is given the statistical patterns the model learned. The model has no mechanism to separate "this token is probable given my training data" from "this claim is true about the world." To build such a distinction would require fundamentally rethinking what these systems output.

One approach would be to have models generate not just predictions but *prediction intervals*—explicit statements about the range of uncertainty around a claim. A medical AI might output: "There is a 78% probability this image contains a tumor, with a calibrated confidence interval of ±12 percentage points, and I have not been trained on images of this specific presentation." A legal research system might return: "This case is analogous to established precedent on the 60% of its facts that match my training data, but I cannot assess the relevance of the 40% I cannot map to prior cases." These outputs would be far less useful for rapid deployment. They would be harder to integrate into workflows. They would slow down decisions. They would also be closer to the truth about what these systems actually know and don't know.

But generating such outputs requires changing the training objective itself. Currently, models are optimized to predict the most likely next token. If you want a model to sometimes output "I don't know" or "this is outside my training distribution" or "I have low confidence in this prediction," you need to reward it for doing so. And this immediately creates a new problem: how do you know when "I don't know" is the honest answer versus when it is an evasion?

This is where the book's diagnosis becomes uncomfortable. You cannot solve this problem by training the model better. You can only solve it by changing what humans do with the model's outputs.

The Institutional Problem

Imagine a hospital implements a new radiology AI system with one strict design principle: the system must communicate uncertainty. It outputs probability distributions instead of binary diagnoses. It flags cases outside its training distribution. When it is unsure, it says so explicitly, and it does so frequently—because it is genuinely unsure about many images, and its training has correctly calibrated it to express that uncertainty.

Now watch what happens in practice.

The radiologists trust it less. The administrators like it less. The workflow is slower. When a radiologist sees "55% confidence: possible small nodule, ±15 percentage points, low confidence due to image quality," they have to make a decision about whether to order follow-up imaging, refer to oncology, or reassure the patient. If the system had said "small nodule detected: 89% confidence," the decision path is clearer, faster, and more deferential to the system's judgment. A system that is honest about its limitations is harder to deploy because it does not allow humans to offload their epistemic responsibility.

This is not cynicism about doctors or hospitals. It is realism about how institutions function at scale. A hospital receiving thousands of imaging studies per day, staffed by radiologists working near capacity, is structurally incentivized to use AI systems that reduce cognitive load. An AI system that generates clear, confident recommendations reduces load. An AI system that generates probabilistic hedging, uncertainty intervals,

and caveats increases load. The institution will prefer the former, even if the latter is more truthful.

The same dynamic appears everywhere these systems are deployed. A legal firm would rather have an AI that confidently identifies relevant precedents than one that constantly qualifies its findings. A school would rather have an educational AI that generates a curriculum with clear learning objectives than one that constantly flags gaps in its understanding. A hiring system that says "I am unsure about this candidate because the training data I was exposed to did not include people from similar demographic backgrounds" is harder to deploy than one that simply ranks candidates by predicted job performance.

The problem is not the technology. It is the institutional environment in which the technology operates. And that environment is not going to change because the models are made more honest. If anything, the pressure points will intensify. As these systems become more economically valuable, the incentive to interpret their outputs charitably, to smooth over their uncertainties, to integrate them into decision-making as quickly as possible, will only grow.

This brings us to the core of what needs to change: not the models, but the relationship between models and decision-making.

Radical Transparency as Institutional Design

What would it mean to build institutions that are structurally resistant to the confidence plague?

The answer is not to make the AI systems better. It is to stop treating them as sources of knowledge in the first place and instead treat them as tools that generate hypotheses for testing.

Consider a radically different approach to medical AI: instead of deploying a system that outputs diagnoses, deploy a system that outputs *testable predictions*. The AI analyzes a patient's presentation and generates not a diagnosis but a ranked list of possible conditions, along with predictions about what tests would disambiguate between them. "Your symptoms are consistent with three patterns in my training data: (1) viral infection, which would show elevated lymphocytes in CBC; (2) autoimmune response, which would show specific antibodies in this panel; (3) early cancer signal, which would show abnormal cells in biopsy. I recommend testing for all three. I am least confident about (3), as it appears in only 2% of my training data with similar presentations." The human clinician is not accepting the AI's diagnosis; they are using it as a hypothesis generator. The AI's uncertainty is not a bug to be fixed; it is information about where to focus investigation.

This requires a different kind of transparency. Not transparency about the model's architecture or training data—that is valuable for researchers but not for practitioners. Rather, transparency about the *scope of what the model can claim to know*. A radiology AI should be explicit about whether it was trained on the same imaging modalities, patient populations, and disease prevalences as the ones it is now being deployed on. A legal research AI should be explicit about whether the cases it found are actually from the relevant jurisdiction and whether it has simply pattern-matched on superficial similarities. A hiring system should be explicit about what demographic groups it was trained on and how its predictions might change for people from groups it has never seen.

Some of this transparency is already being pursued under the umbrella of "model cards" and "system cards"—documenting not just how a model performs on average but how it performs across different slices of data. This is progress. But it remains largely a research and compliance practice.

In actual deployment, the cards exist in documentation no one reads, posted before the system is shipped to production where the pressure to keep it running overrides the caution it was supposed to generate.

Real transparency would be built into the deployment itself. It would be structural and unavoidable. It would mean that whenever a system outputs a prediction, it also outputs metadata about that prediction: the confidence interval, the degree to which this case resembles the training data, the specific training subsets it is most similar to, the false positive and false negative rates on those subsets, and the base rates of the condition in the populations the system was trained on.

This would make systems harder to use. It would slow deployment. It would also force the humans in the decision loop to actually think about what the system is telling them rather than simply accepting it.

There is a name for institutions that are structured to resist overconfidence and to require constant engagement with uncertainty: they are called good science.

The Science Analogy and Its Limits

Science does not work by producing confident predictions and deploying them immediately. It works by generating hypotheses, testing them against reality, failing often, and revising understanding based on falsification. A scientific claim is only as good as the evidence that supports it, and that evidence is never final—it is always subject to new data, new methods, and new scrutiny.

Most AI systems in deployment are structured nothing like this. They are optimized to produce predictions quickly, integrated into decision-making workflows where humans lack the time and information to verify them, and

updated only when they clearly fail in ways that are impossible to ignore. They are as far from scientific practice as institutions can get while still claiming to be evidence-based.

What if we designed AI systems to operate within scientific workflows rather than to replace human judgment?

In this model, AI generates hypotheses, and humans design experiments to test them. A medical AI might suggest that a patient's symptoms are consistent with a rare condition, and the clinician designs a test to falsify or confirm that hypothesis. A legal AI might identify a pattern of prior cases, and the attorney works to find cases that contradict that pattern or clarify its boundaries. An educational AI might predict that a student would benefit from a different learning approach, and the teacher tries it and observes the results.

This requires the humans in the system to be actively engaged in evaluation, not passive recipients of the system's outputs. It requires them to have the time, expertise, and institutional support to push back on the system's recommendations. And it requires the system itself to be designed as a tool for generating candidates for investigation rather than as an oracle delivering final answers.

The obstacle to this is not technical. It is organizational and economic. Building such workflows is slower and more expensive than deploying systems that users can treat as black-box decision-makers. The entire business model of AI in deployment relies on efficiency—on systems that can process more data with fewer humans. A workflow where AI and human expertise are deeply integrated, where the system's recommendations are constantly tested and revised, where uncertainty is treated as information rather than as a bug to be engineered away—that

workflow scales differently. It might be cheaper per unit of actual correctness, but it looks more expensive upfront.

This is where the problem becomes genuinely difficult. Because the solution is not "make better AI." It is "restructure institutions to value correctness over efficiency, and create feedback loops that reveal when you are wrong." And institutions are reluctant to do this because they are already optimized for speed.

What Has to Break

For the mimic's problem to be genuinely addressed, several things would need to change simultaneously, and none of them are changes to the models themselves.

First, incentive structures would need to shift toward systems that generate falsifiable predictions rather than authoritative ones. A medical AI that says "this warrants follow-up imaging" is better than one that says "this is definitely cancer" because the former can be tested against reality. The latter requires months of follow-up to disprove and is designed to foreclose further investigation. We would need to reward systems and the organizations deploying them for generating predictions that are specific, testable, and open to falsification—and to penalize systems that trade specificity for false confidence.

Second, deployment would need to include mandatory ground-truth feedback loops. Right now, many AI systems in production never learn whether they were right or wrong. A hiring system does not systematically track the job performance of candidates it rejected. A medical system does not correlate its diagnoses with patient outcomes. This is partly because ground truth is expensive to collect, but partly because there is no

institutional pressure to collect it. If we required that any AI system making decisions affecting humans also maintained a dataset of ground-truth outcomes and used that to continuously audit its own performance, the gap between plausibility and correctness would become visible. The system would be forced to learn where it was failing, or it would be taken offline.

Third, the training objective itself would need to change. We have optimized AI systems to predict the most likely next token, the most probable category, the highest-scoring candidate. But maximizing likelihood is not the same as maximizing correctness. A system that is trained to express uncertainty when appropriate—even when expressing uncertainty lowers its score on standard metrics—would look different. It would give fewer confident answers. It would say "I don't know" more often. And by most conventional measures, it would look worse. We would need to build organizations and evaluation frameworks that reward it anyway.

Fourth, humans in the decision loop would need the power to push back on the system's recommendations. This sounds obvious, but it is not how these systems are typically deployed. They are deployed in contexts of time pressure and information asymmetry, where the human decision-maker lacks the expertise or time to evaluate the system's reasoning. A radiologist examining 100 scans a day cannot carefully verify the AI's diagnosis on each one. A hiring manager reviewing 500 applications cannot deep-dive into why the system ranked each candidate. The system's output becomes the de facto decision. To genuinely integrate human judgment with machine-generated hypotheses requires building workflows where the human has the time, expertise, and authority to disagree with the system. This is expensive. It does not scale as easily. And it is what responsible AI deployment would actually look like.

These four changes are not independent. Each one only works if the others are also in place. You cannot get feedback if the system is optimized to avoid making falsifiable claims. You cannot change the training objective if the institution that benefits from the system's current behavior does not want it to change. You cannot give humans the power to evaluate the system if they lack the expertise or time.

The problem, in other words, is systemic. And systemic problems do not have quick fixes.

The Honesty Constraint

There is one more piece that might be worth saying, because it cuts against much of the optimism in AI circles.

Building AI systems that are honest about their limitations would make them less impressive in the ways that matter for funding, hype, and adoption. A system that constantly qualifies its outputs, that expresses uncertainty, that flags the limits of what it knows, does not perform as well on benchmarks. It does not generate the flashy demos that attract investors. It does not reduce costs as dramatically in the initial deployment phase. It does not satisfy the organizational desire to automate away the human judgment that is expensive and variable and unpredictable.

For this reason, the incentive to build such systems is weak. And the incentive to deploy them, once built, is even weaker. An organization that implements an AI system is accountable for the system's performance. If the system says "I'm not sure" and a human acts on that uncertainty and something goes wrong, the human might be blamed. If the system says "I'm sure" and something goes wrong, the organization can point to the system and say it failed, and move to fix or replace it. The system becomes a kind

of institutional shield, and the more authoritative it sounds, the more effectively it shields.

This is not a problem that can be solved by asking companies to be more ethical. Ethics frameworks, responsible AI principles, and alignment-focused research are all valuable, but they operate within a structure that is fundamentally opposed to them. An organization optimized for efficiency and liability minimization will not voluntarily adopt workflows that are less efficient, even if those workflows are more truthful, because the structure does not reward truthfulness—it rewards speed and legal defensibility.

What would actually need to change is the liability structure itself. If organizations were held accountable not just for obviously false claims but for claims that were confident beyond what their data warranted, they would have reason to build systems differently. If there were legal consequences for deploying AI in high-stakes decisions without adequate ground-truth feedback loops, deployment practices would change. If false confidence were treated not as a deployment risk to be managed but as a fundamental violation of duty, institutions would structure their use of AI differently.

This sounds like regulation. In some sense, it is. But not the kind of regulation that typically gets proposed for AI—restrictions on model size, or algorithmic audits, or diversity requirements in training data. Rather, it would be structural regulation of how these systems can be deployed in consequential domains. It would be less about making the models better and more about constraining their use to contexts where their limitations can be reliably exposed.

Some of this is already beginning. Medical AI systems are increasingly required to report confidence intervals. Legal firms are being trained on the

risks of AI hallucination. Educational institutions are starting to acknowledge that AI-generated content might be plausible without being true. But these changes are still upstream of the actual decision-making, still treated as problems to manage rather than as constraints that fundamentally restructure how the systems can be used.

What Comes After

This book opened by documenting the confidence plague—the way AI systems fail not by being uncertain but by being wrongly confident. It traced that plague to its source: the fundamental mismatch between optimizing for plausibility and optimizing for truth. And it showed how that mismatch becomes invisible when systems are deployed in contexts where ground truth is rare and feedback is delayed.

The question of how to fix this has been lurking throughout. And the answer, stated plainly, is that the fix is not to the systems themselves. It is to the institutions that deploy them and the humans who use them and the structures of accountability that determine what happens when they fail.

An AI system that is honest about its limitations is possible to build. It would be less impressive, less profitable, and slower to deploy than current systems. It would require humans to do more cognitive work rather than less. It would reveal uncertainty rather than smoothing it over. And it would be, by most measures that matter for institutional adoption, worse.

This is the real insight: the mimic's problem is not a problem of intelligence or understanding or alignment. It is a problem of what we have chosen to reward.

We built systems that are very good at looking like they know what they are talking about, and then we deployed them in contexts where that

appearance is rewarded regardless of its correspondence to reality. We trained them to optimize for plausibility, and then we installed them in decision-making workflows where plausibility closes the loop—where the system's confident output becomes the input that shapes human behavior, which becomes the training signal that confirms the system's confidence.

What comes after the mimic is the choice to stop building that loop. It means accepting that AI systems are tools for generating candidates for investigation, not oracles for dispensing decisions. It means demanding that institutions maintain contact with ground truth, that they measure what actually happened rather than what the system predicted would happen. It means being willing to slow down, to preserve space for human judgment, to tolerate uncertainty in the name of something closer to correctness.

It means, in other words, choosing to build institutions that value truth over speed, even when truth is slower and more expensive.

Whether we will make that choice is not a question for AI researchers or engineers. The technology itself is not the constraint. The constraint is whether we are willing to restructure the economic and institutional incentives that made the mimic profitable in the first place. And that is a question about what we actually care about, which is the hardest kind of question to answer.